

Because of the outdated latex system of arxiv,  
this is a less-featured version of my diploma  
theses.

Please see <http://robert.bredereck.info> for the  
full-featured version.

Friedrich-Schiller-Universität Jena



seit 1558

## DIPLOMARBEIT

# Graph and Election Problems Parameterized by Feedback Set Numbers

zur Erlangung des akademischen Grades  
Diplom-Informatiker

ausgeführt am Lehrstuhl für Theoretische Informatik I / Komplexitätstheorie  
an der Fakultät für Mathematik und Informatik  
der Friedrich-Schiller-Universität Jena

unter Anleitung von Dipl.-Bioinf. Nadja Betzler, Dipl.-Bioinf. Johannes Uhlmann  
und Prof. Dr. Rolf Niedermeier

durch

Robert Bredereck  
geb. am 31.01.1985 in Wolgast

Jena, 25. Mai 2010

# Abstract

This work investigates the parameterized complexity of three related graph modification problems. Given a directed graph, a distinguished vertex, and a positive integer  $k$ , MINIMUM INDEGREE DELETION asks for a vertex subset of size at most  $k$  whose removal makes the distinguished vertex the only vertex with minimum indegree. MINIMUM DEGREE DELETION is analogously defined, but deals with undirected graphs. BOUNDED DEGREE DELETION is also defined on undirected graphs, but has a positive integer  $d$  instead of a distinguished vertex as part of the input. It asks for a vertex subset of size at most  $k$  whose removal results in a graph in which every vertex has degree at most  $d$ . The first two problems have applications in computational social choice whereas the third problem is used in computational biology. We investigate the parameterized complexity with respect to the parameters “treewidth”, “size of a feedback vertex set” and “size of a feedback edge set” respectively “size of a feedback arc set”. Each of these parameters measures the “degree of acyclicity” in different ways. For MINIMUM INDEGREE DELETION we show that it is W[2]-hard with respect to both parameters that are defined on acyclic graphs. We describe a branch-and-bound algorithm whose running time is  $O(s \cdot (k + 1)^s \cdot n^2)$ , where  $n$  is the number of vertices,  $k$  is the “number of vertices to delete”, and  $s$  is the “size of a feedback set”. For MINIMUM DEGREE DELETION we show W[1]-hardness with respect to the parameter “number of vertices to delete”. With respect to each of the parameters that measures the “degree of acyclicity” we show fixed-parameter tractability. We describe a simple search tree algorithm with running time  $O(2^s \cdot n^3)$  where  $n$  is the number of vertices and  $s$  is the “size of a feedback edge set” and two concrete fixed-parameter algorithms with respect to the parameter “size of a feedback vertex set that does not contain the distinguished vertex”. For BOUNDED DEGREE DELETION we present a search-tree algorithm with running time  $O(3^s \cdot n^2)$  where  $n$  is the number of vertices and  $s$  is the “size of a feedback edge set”.

# Zusammenfassung

Diese Arbeit untersucht die parametrisierte Komplexität von drei verwandten Graphmodifikationsproblemen. Jedes dieser Probleme sucht eine Knotenteilmenge beschränkter Größe, deren Löschung zu einem Graph mit einer problemspezifischen Eigenschaft führt. Das erste betrachtete Problem MINIMUM INDEGREE DELETION hat als Eingabe einen gerichteten Graphen, einen ausgewiesenen Knoten und eine natürliche Zahl  $k$ . Die Frage ist, ob der ausgewiesene Knoten durch Löschung von maximal  $k$  Knoten der einzige Knoten mit minimalem Eingangsgrad werden kann. Das Problem MINIMUM INDEGREE DELETION wurde im Kontext der Wahlforschung eingeführt. Genauer gesagt analysiert man für Wahlsysteme ein „Control“-Szenario, in welchem man fragt, ob durch Löschung einer durch die Größe beschränkten Kandidatenmenge das Ergebnis so zu beeinflussen ist, dass ein ausgewählter Kandidat gewinnt. Die Ergebnisse bezüglich MINIMUM INDEGREE DELETION lassen sich eins zu eins auf dieses Szenario übertragen. Das zweite betrachtete Problem MINIMUM DEGREE DELETION ist Analog zu MINIMUM INDEGREE DELETION auf ungerichteten Graphen definiert. Hier fordert man, dass nach Löschung der Knotenmenge ein ausgewählter Knoten als einziger minimalen Grad hat. Auch dieses Problem lässt sich durch ein natürliches, auf sozialen Netzwerken basierendes Wahlsystem motivieren. Das dritte betrachtete Problem BOUNDED DEGREE DELETION ist ebenfalls auf ungerichteten Graphen definiert. Es hat jedoch im Gegensatz zu den beiden vorherigen Problemen keinen ausgewiesenen Knoten als Teil der Eingabe sondern fordert, dass nach Löschung von maximal  $k$  Knoten alle Knoten maximal Grad  $d$  besitzen für ein bestimmtes zur Eingabe gehöriges  $d$ . Das Problem BOUNDED DEGREE DELETION findet Anwendung bei der Analyse von biologischen Netzwerken.

Neben Ähnlichkeiten in der Definition haben die drei betrachteten Probleme gemeinsam, dass sie auf kreisfreien Graphen in Polynomzeit lösbar sind, während sie jedoch im Allgemeinen NP-schwer sind. Diese Eigenschaft war Ausgangspunkt für die Untersuchung der parametrisierten Komplexität bezüglich drei verschiedener Parameter, welche jeweils auf unterschiedliche Weise die Distanz des Eingabegraphen zu einem kreisfreien Graphen messen. Der erste dieser Parameter ist die für die Untersuchung ungerichteter Graphen weit verbreitete „Baumweite“. Die beiden anderen Parameter messen jeweils die Anzahl der Knoten bzw. Kanten deren Löschung zu einem kreisfreien Graphen führt. Genauer gesagt sind dies die Parameter „Größe einer kreiskritischen Knotenmenge“ und „Größe einer kreiskritischen Kantenmenge“.

Für MINIMUM INDEGREE DELETION zeigen wir, dass es bezüglich beider auf gerichteten Graphen definierten Parameter W[2]-schwer und damit offenbar nicht festparameterhandhabbar ist. Betrachtet man hingegen den Parameter „Anzahl zu löschender Knoten“ in Kombination mit einem der beiden anderen, lässt sich Festparameterhandhabbarkeit nachweisen. Wir beschreiben einen einfachen Algorithmus dessen Laufzeit  $O(s \cdot (k + 1)^s \cdot n^2)$  ist, wobei  $n$  die Anzahl der Knoten,  $k$  die „Größe der zu löschenden Knotenmenge“ und  $s$  die „Größe einer kreiskritischen Menge“ ist. Für MINIMUM DEGREE DELETION wird zusätzlich der Parameter „Anzahl zu löschender Knoten“ betrachtet und für diesen W[1]-Schwere nachgewiesen. Bezüglich jedem der Parameter „Baumweite“, „Größe einer kreiskritischen Knotenmenge“ sowie „Größe einer kreiskritischen Kantenmenge“ wird Festparameterhandhabbarkeit gezeigt. Wir zeigen bezüglich „Größe einer kreiskritischen Kantenmenge“ einen Problemkern, dessen Größe linear in der Anzahl der Knoten ist. Im Gegensatz dazu zeigen wir bezüglich der anderen beiden Parameter, dass es keinen Problemkern polynomieller Größe gibt, es sei denn, die Polynomialzeithierar-

chie kollabiert. Wir beschreiben einen Suchbaumalgorithmus mit Laufzeit  $O(2^s \cdot n^3)$ , wobei  $n$  die Anzahl der Knoten und  $s$  die „Größe einer kreiskritischen Kantenmenge“ ist. Zwei konkrete Festparameteralgorithmen werden bezüglich des Parameters „Größe einer kreiskritischen Knotenmenge, welche nicht den ausgewiesenen Knoten enthält“ präsentiert. Wir weisen abschließend für BOUNDED DEGREE DELETION Festparameterhandhabbarkeit bezüglich des Parameters „Größe einer kreiskritischen Kantenmenge“ nach. Ein entsprechender Algorithmus hat die Laufzeit  $O(3^s \cdot n^2)$ , wobei  $n$  die Anzahl der Knoten und  $s$  die „Größe einer kreiskritischen Kantenmenge“ ist.



# Contents

<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Preliminaries . . . . .	11
1.2	Parameterized complexity . . . . .	12
<b>2</b>	<b>Degree-based vertex deletion problems</b>	<b>17</b>
2.1	Minimum Indegree Deletion . . . . .	17
2.2	Minimum Degree Deletion . . . . .	18
2.3	Bounded Degree Deletion . . . . .	19
<b>3</b>	<b>Feedback sets</b>	<b>21</b>
<b>4</b>	<b>Minimum Indegree Deletion</b>	<b>25</b>
4.1	Known results . . . . .	25
4.2	Feedback vertex/arc set size as parameter . . . . .	26
4.3	Feedback vertex set size and solution size as combined parameter . . . . .	31
<b>5</b>	<b>Minimum Degree Deletion</b>	<b>35</b>
5.1	Solution size as parameter . . . . .	36
5.2	Size of a feedback edge set as parameter . . . . .	39
5.3	Treewidth as parameter . . . . .	42
5.3.1	Monadic second-order logic . . . . .	43
5.3.2	MSO expression for Minimum Degree Deletion . . . . .	45
5.4	Size of a feedback vertex set as parameter . . . . .	47
5.4.1	Integer linear programming . . . . .	49
5.4.2	Dynamic programming . . . . .	53
5.5	No polynomial kernel with respect to $s_v$ . . . . .	57
<b>6</b>	<b>Bounded Degree Deletion</b>	<b>63</b>
6.1	Known results . . . . .	63
6.2	Size of a feedback edge set as parameter . . . . .	63
<b>7</b>	<b>Conclusion and Outlook</b>	<b>69</b>





# 1 Introduction

This work investigates the parameterized complexity of three related graph modification problems. Each of these problems asks for a vertex subset of bounded size whose removal results in a graph that satisfies a problem-specific property. The first considered problem MINIMUM INDEGREE DELETION has a directed graph, a distinguished vertex, and a positive integer  $k$  as input. The question is whether the distinguished vertex can be made the only vertex with minimum indegree by removing at most  $k$  vertices. The problem MINIMUM INDEGREE DELETION was introduced in the context of computational social choice [BU09]. More precisely, one analyzes a so-called “control” scenario for a voting rule in which one asks to manipulate the result of the voting rule by removing a size-bounded candidate subset such that a distinguished candidate becomes the only winner. The results regarding MINIMUM INDEGREE DELETION can be transferred to this scenario [BU09]. The second considered problem MINIMUM DEGREE DELETION is defined analogously to MINIMUM INDEGREE DELETION, dealing with undirected graphs. Here one asks whether it is possible to make a distinguished vertex the only vertex with minimum degree by removing a subset of vertices. This problem can be motivated by a simple voting rule that is based on a social network. The third considered problem BOUNDED DEGREE DELETION [Mos10] is also defined on undirected graphs, but in contrast to both previous problems it has no distinguished vertex as part of the input. It asks for a set of  $k$  vertices whose removal results in a graph with maximum degree  $d$  for a specific positive integer  $d$  that is part of the input. The problem BOUNDED DEGREE DELETION has applications in the analysis of genetic networks [Mos10]. In Chapter 2 we provide formal definitions and more details on applications.

Besides similarities in the definition, all three problems have in common that they are solvable in polynomial time on acyclic graphs, but NP-hard in general. This property is the starting point for the investigation of the parameterized complexity with respect to three distinct parameters which measure the distance of the input graph to an acyclic graph in three different ways. The first of these parameters is the “treewidth”, being well-known in the analysis of undirected graphs. The two other parameters measure in each case the number of vertices respectively edges whose removal results in an acyclic graph. More precisely, we have the parameters “size of a feedback vertex set” and “size of a feedback edge set” respectively “size of a feedback arc set” in the directed case. A formal definition of “treewidth” is given in Section 1.1 whereas the other parameters are considered in detail in Chapter 3.

In what follows, we briefly describe our results, also see Figure 1.1. For MINIMUM INDEGREE DELETION we show that it is  $W[2]$ -hard, hence, that it is presumably not fixed-parameter tractable with respect to both parameters that are defined on acyclic graphs (see Section 4.2). Considering the parameter “number of vertices to delete”

parameter	MID	MDD	BDD
$t_w$		<b>FPT</b>	open
$s_v$	<b>W[2]-hard, in XP</b>	<b>FPT</b>	open
$s_{a/e}$	<b>W[2]-hard, in XP</b>	<b>FPT</b>	<b>FPT</b>
$k$	W[2]-complete	<b>W[1]-hard, in XP</b>	W[2]-complete
$d$	FPT	FPT	FPT
$(s_v, k)$	<b>FPT</b>	<b>FPT</b>	open

	parameter description
$t_w$	treewidth of the input graph
$s_v$	size of a feedback vertex set
$s_{a/e}$	size of a feedback arc/edge set
$k$	size of a solution set
$d$	maximum degree of a vertex

Figure 1.1: Overview of the parameterized complexity of MINIMUM INDEGREE DELETION (MID), MINIMUM DEGREE DELETION (MDD) and BOUNDED DEGREE DELETION (BDD). New results are in boldface. The remaining results are obtained from [BU09] and [Mos10].

in combination with one of both parameters, we show fixed-parameter tractability. We describe a branch-and-bound algorithm whose running time is  $O(s \cdot (k + 1)^s \cdot n^2)$ , where  $n$  is the number of vertices,  $k$  is the “number of vertices to delete”, and  $s$  is the “size of a feedback set” (see Section 4.3). For MINIMUM DEGREE DELETION we additionally consider the single parameter “number of vertices to delete” and show W[1]-hardness (see Section 5.1). With respect to each of the parameters “treewidth”, “size of a feedback vertex set”, and “size of a feedback edge set” we show fixed-parameter tractability (see Section 5.3.2). We show with respect to “size of a feedback edge set” a problem kernel whose size is linear in the number of vertices (see Section 5.2). In contrast, we show that with respect to each of the two other parameters there is no problem kernel of polynomial size, assuming that the polynomial-time hierarchy does not collapse (see Section 5.5). We describe a simple search tree algorithm with running time  $O(2^s \cdot n^3)$  where  $n$  is the number of vertices and  $s$  is the “size of a feedback edge set”. This algorithm complements the kernelization result from Section 5.2. We present two concrete fixed-parameter algorithms with respect to the parameter “size of a feedback vertex set that does not contain the distinguished vertex”. The first one uses the technique of integer linear programming (see Section 5.4.1) whereas the second one is a dynamic programming algorithm (see Section 5.4.2). Besides the intractability results in Section 4.2 and Section 5.1, the two algorithms with respect to the parameter “size of a feedback vertex set that does not contain the distinguished vertex” are the most demanding parts of this work. Finally, we show fixed-parameter tractability for BOUNDED DEGREE DELETION with respect to “size of a feedback edge set”. A corresponding search-tree algorithm has running time  $O(3^s \cdot n^2)$  where  $n$  is the number of vertices and  $s$  is the “size of a feedback edge set”.

**Organization of this work.** The second chapter gives an overview of the considered problems and its applications. A third chapter contains some definitions and computational aspects of the considered parameters measuring the distance of the input graph from an acyclic graph. Each of the Chapters 4-6 investigates one of the considered vertex deletion problem. The last chapter provides a conclusion and outlooks to further investigations in this field.

## 1.1 Preliminaries

We introduce the basic terms and methods which are necessary in this work.

**Graph theory.** All computational problems that are investigated in this work are graph problems. An *undirected graph*  $G$  is a pair  $(V, E)$ , where  $V$  is a finite set of *vertices* and  $E$  is a finite set of *edges* which are unordered pairs of vertices. A *directed graph*  $G$  is also a pair  $(V, A)$ , where  $V$  is a finite set of vertices, but  $A$  is a finite set of *arcs* which are ordered pairs of vertices. All graphs considered in this work are *simple*, that is, there are no multi-arc/multi-edges, and they do not contain *self-loops*, that is, no edge/arc from a vertex to itself. Let  $\{v_1, v_2\}$  be an edge of an undirected graph. Then,  $v_1$  is a *neighbor* of  $v_2$  and vice versa. We denote  $\deg(x)$  as the *degree* of the vertex  $x$ , that is, the number of its neighbors. Furthermore, the (open) *neighborhood* of a vertex  $v$  in an undirected graph  $G = (V, E)$  is defined as  $N(v) := \{u \mid \{u, v\} \in E\}$ . Let  $(v_1, v_2)$  be an arc of a directed graph. We denote  $v_1$  as *inneighbor* of  $v_2$  and  $v_2$  as *outneighbor* of  $v_1$ . The *indegree* of a vertex is the number of its inneighbors and the *outdegree* of a vertex is the number of its outneighbors. The (open) *inneighborhood* of a vertex  $v$  in a directed graph  $G = (V, A)$  is defined as  $N_{\text{in}}(v) := \{u \mid (u, v) \in A\}$  and the (open) *outneighborhood* of a vertex  $v$  in a directed graph  $G = (V, A)$  is defined as  $N_{\text{out}}(v) := \{u \mid (v, u) \in A\}$ . For a vertex set  $S \subseteq V$ , we write  $G[S]$  to denote the graph induced by  $S$  in  $G$ , that is,  $G[S] := (S, \{e \in E \mid e \subseteq S\})$  for an undirected graph  $G = (V, E)$  respectively  $G[S] := (S, \{(x, y) \in A \mid x \in S \wedge y \in S\})$  for a directed graph  $G = (V, A)$ . For a subset  $S \subseteq V$ , we also write  $G - S$  instead of  $G[V \setminus S]$ . We define  $P_i := (V_{P_i} := \{v_1, \dots, v_i\}, E_{P_i} := \{\{a, b\} \mid 1 \leq a < b \leq i\})$  as *path of length  $i$*  between  $v_1$  and  $v_i$ . Moreover,  $C_i := (V_{P_i}, E_{P_i} \cup \{v_i, v_1\})$  is defined as *cycle of length  $i$* . A graph that contains a cycle as subgraph is called *cyclic*. Otherwise we say the graph is *acyclic*. In directed graphs the terms *path of length  $i$*  ( $P_i := (V_{P_i} := \{v_1, \dots, v_i\}, E_{P_i} := \{(a, b) \mid 1 \leq a < b \leq i\})$ ), *cycle of length  $i$*  ( $C_i := (V_{P_i}, E_{P_i} \cup (v_i, v_1))$ ), and *cyclic/acyclic* are defined analogously. An undirected graph is *connected* if there is a path between each two vertices. A connected and acyclic graph is called a *tree*.

**Tree decomposition and treewidth.** As mentioned in the introduction, many hard graph problems are easy when restricted on acyclic graph. The main question is: “Why is an NP-hard problem in P when restricted on a tree?”. We need a measure of the “tree-likeness” of a given graph. Robertson and Seymour [RS86] introduced the concept of tree-decompositions and treewidth of undirected graphs.

**Definition 1.** Let  $G = (V, E)$  be an undirected graph. A **tree decomposition** of  $G$  is a pair  $(\{X_i \mid i \in I\}, T)$  where each  $X_i$  is a subset of  $V$ , called a **bag**, and  $T$  is a tree with the elements of  $I$  as nodes. The following three properties must hold:

1.  $\bigcup_{i \in I} X_i = V$ ,
2. for every edge  $\{u, v\} \in E$ , there is an  $i \in I$  such that  $\{u, v\} \subseteq X_i$ , and
3. for all  $i, j, k \in I$ , if  $j$  lies on the path between  $i$  and  $k$  in  $T$ , then  $X_i \cap X_k \subseteq X_j$ .

The **width** of the tree decomposition  $(\{X_i \mid i \in I\}, T)$  equals  $\max\{|X_i| \mid i \in I\} - 1$ . The **treewidth** of  $G$  is the minimum  $k$  such that  $G$  has a tree decomposition of width  $k$ .

Clearly, a (non-trivial) tree has treewidth 1, a cycle has treewidth 2, and a complete graph with  $n$  vertices has treewidth  $n - 1$ . The other two parameters which measure the “degree of acyclicity” are introduced in Chapter 3.

## 1.2 Parameterized complexity

Many interesting problems in computer science are computationally hard problems in worst case. The most famous class of such hard problems is the class of NP-hard problems. The relation between P (which includes the “efficient solvable problems”) and NP is not completely clear at the moment. Even if  $P = NP$  it is not self-evident that we are able to design *efficient* polynomial-time algorithms for each NP-hard problem. But we have to solve NP-hard problems in practice. Thus, according to the state of the art of computational complexity theory, NP-hardness means that we only have algorithms with exponential running times to solve the corresponding problems exactly. This is a huge barrier for practical applications. There are different ways to cope with this situation: heuristic methods, randomized algorithms, average-case analysis (instead of worst-case) and approximation algorithms. Unfortunately, none of these methods provides an algorithm that efficiently computes an optimal solution in the worst case. Since there are situations where we need performance and optimality guarantee at least for a specified type of input, another way out is needed. Fixed-parameter algorithms provide a possibility to redefine problems with several input parameters. The main idea is to analyze the input structure to find parameters that are “responsible for the exponential running time”. The aim is to find such a parameter, whose values are constant or “logarithmic in the input size” or “usually small enough” in the problem instances of your application. Thus, we can say something like “if the parameter is small, we can solve our problem instances efficiently”.

We will use the two-dimensional parameterized complexity theory [DF99, Nie06, FG06] for studying the computational complexity of several graph problems. A *parameterized problem* (or language)  $L$  is a subset  $L \subseteq \Sigma^* \times \mathbb{N}$  for some finite alphabet  $\Sigma$ . For an element  $(x, k)$  of  $L$ , by convention  $x$  is called *problem instance*<sup>1</sup>

<sup>1</sup>Most parameterized problems originate from classical complexity problems. One can interpret  $x$  as the input of the original/non-parameterized problem.

and  $k$  is the *parameter*. The two dimensions of parameterized complexity theory are the size of the input  $n := |(x, k)|$  and the parameter value  $k$ , which is usually a non-negative integer. A parameterized language is called *fixed-parameter tractable* if we can determine in  $f(k) \cdot n^{O(1)}$  time whether  $(x, k)$  is an element of our language, where  $f$  is a computable function only depending on the parameter  $k$ . The class of fixed-parameter tractable problems is called FPT. Thus, it is very important to find good parameters. In the following chapters, we need four of the core tools in the development of parameterized algorithms [Nie06]: data reduction rules (kernelization), (depth-bounded) search trees, dynamic programming and integer linear programs.

The idea of *kernelization* is to transform any problem instance  $x$  with parameter  $k$  in polynomial time into a new instance  $x'$  with parameter  $k'$  such that the size of  $x'$  is bounded from above by some function only depending on  $k$  and  $k' \leq k$ , and  $(x, k) \in L$  if and only if  $(x', k') \in L$ . The reduced instance  $(x', k')$  is called *problem kernel*. This is done by *data reduction rules*, which are transformations from one problem instance to another. A data reduction rule that transforms  $(x, k)$  to  $(x', k')$  is called *sound* if  $(x, k) \in L$  if and only if  $(x', k') \in L$ .

Besides kernelization we use (depth-bounded) *search trees algorithms*. A search algorithm takes a problem as input and returns a solution to the problem after evaluating a number of possible solutions. The set of all possible solutions is called the search space. Depth-bounded search tree algorithms organize the systematic and exhaustive exploration of the search place in a tree-like manner. Let  $(x, k)$  denote the instance of a parameterized problem. The search tree algorithm replaces  $(x, k)$  by a set  $H$  of smaller instances  $(x_i, k_i)$  with  $|x_i| < |x|$  and  $k_i < k$  for  $1 \leq i \leq |H|$ . Thus, the search tree size (number of nodes) is clearly bounded by  $|H|^k$ . Since the running time of the replacement procedure is bounded by a polynomial in the instance size, a constant-size set  $H$  always leads to a fixed-parameter algorithm with respect to  $k$ . However, there are more refined methods to compute a better upper bound for the search tree size using the so-called branching vector, but they are not important in this work.

Another important technique used in this work is *dynamic programming*. It goes back to Bell [Bel03]. As well as search trees dynamic programming makes exhaustive search, but is more efficient by avoiding the computation of subproblems more than once. It is used when a problem exhibits the property of having *optimal substructure*, that is, an optimal solution to the problem contains within it optimal solutions to subproblems. Two further properties are important for a feasible dynamic programming: *Independence*, that is, the solution of one subproblem does not affect the solution of another subproblem of the same problem, and *overlapping subproblems*, that is, the same problem occurs as a subproblem of different problems. One organizes the computation of the solutions for the subproblems in the *dynamic programming table*. An established way to compute the table which is used in this work is the *bottom-up* computation. Typically in fixed-parameter algorithms, the computation of a single table entry depends only on a constant number of parent entries and can be done in polynomial time. The table size is bounded by a function only depending on the parameter. Furthermore, the overall-solution can be computed from the completely filled table in polynomial time.

The technique of *integer linear programming* is a special case of *linear programming* which goes back to Kantorovich [Kan60]. It is a technique for the optimization (minimization and maximization) of a (linear) *objective function*, subject to linear equality and linear inequality constraints. In an *integer linear program* (ILP), the unknown variables are required to be integers instead of real number as in a linear program. Whereas solving a linear program is possible in polynomial time, integer linear programs are NP-hard [Kar72]. Anyhow, the technique can be used to show fixed-parameter tractability when the number of unknown variables is bounded by a function only depending on the parameter [Len83, Kan87]. More about these four techniques can be found in [Nie06].

In many applications one is interested in deciding an NP-hard problem or computing the optimal solution of the corresponding search or optimization problem. Therefore, fixed-parameter tractability is a desired attribute of a problem together with a well-chosen parameter. However, there are also some cases where intractability can also be a desired attribute. For example voting rules seem to be “more fair” if a (at this point not more specified) manipulation or control of that rule is computationally hard. Thus, parameterized intractability can be a positive result as well as negative result. In this work, we use a characterization of parameterized problems that provides even more than only determining fixed-parameter tractability or intractability. In analogy to the concepts of NP-hardness, NP-completeness, and polynomial-time many-to-one reductions in classical complexity theory, Downey and Fellows [DF99] developed a framework of reductions and a hierarchy of parameterized complexity classes.

**Definition 2.** A *parameterized reduction* from a parameterized problem  $L \subseteq \Sigma^* \times \mathbb{N}$  to another parameterized problem  $L' \subseteq \Sigma^* \times \mathbb{N}$  is a function that, given an instance  $(I, k)$ , returns in time  $f(k) \cdot \text{poly}(|(I, k)|)$  an instance  $(I', k')$ , with  $k'$  only depending on  $k$ , such that  $(I, k) \in L$  if and only if  $(I', k') \in L'$ .

A parameterized problem  $L$  belongs to  $W[t]$  if  $L$  can be reduced to a weighted satisfiability problem for the family of circuits of depth at most some function only depending on the parameter and width at most  $t$ , where width is the maximum number of gates with unrestricted fan-in on an input-output path in the circuit. Similar to the classical complexity theory we denote a problem as  $W[t]$ -hard if there is a parameterized reduction from a problem that is already known to be  $W[t]$ -complete. In this work, the most important thing we have to know about the  $W[t]$ -hierarchy is that every parameterized problem which is  $W[t]$ -hard for  $t \geq 1$  is believed to be not fixed-parameter tractable. (This holds under the separation hypothesis  $\text{FPT} \neq W[1]$ .) A parameterized problem  $L$  belongs to the class XP if it can be determined in  $f(k) \cdot |x|^{g(k)}$  time whether  $(x, k) \in L$  where  $f$  and  $g$  are computable functions only depending on the parameter  $k$ . It holds that

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq \dots \subseteq \text{XP}.$$

More details about this fields can be found in [DF99, FG06, Nie06]. An established way of proving  $W[t]$ -hardness is to give a parameterized reduction from a problem that is already known to be  $W[t]$ -hard. In this work, we use the following two problems:

**INDEPENDENT SET**

*Given:* An undirected graph  $G = (V, E)$  and an integer  $k \geq 1$ .

*Question:* Is there a subset  $V' \subseteq V$  of size at least  $k$  such that there are no edges in  $G[V']$ ?

**DOMINATING SET**

*Given:* An undirected graph  $G = (V, E)$  and an integer  $k \geq 1$ .

*Question:* Is there a subset  $V' \subseteq V$  of size at most  $k$  such that every vertex in  $V$  either belongs to  $V'$  or has a neighbor in  $V'$ ?

The INDEPENDENT SET problem with respect to the parameter  $k$  is known to be W[1]-complete and the DOMINATING SET problem with respect to the parameter  $k$  is known to be W[2]-complete [DF99].





## 2 Degree-based vertex deletion problems

In this work, we analyze three quite similar graph problems. All problems get a directed or undirected graph as one part of the input. In each problem one asks to delete a subset of vertices of bounded size to get a modified graph that satisfies a problem-specific property. Furthermore, this property depends in each of the three problems on the degree of the vertices. However, the particular problems have quite different applications. In the following three sections, we give a formal definition and a motivation (“Why is this problem relevant?”).

### 2.1 Minimum Indegree Deletion

Recently, social choice problems became important in the fields of computational complexity and algorithmics. In this context, the investigation of voting systems is a relevant area. There are two recent surveys by Chevalerey et al. [CELM07] and Faliszewski et al. [FHHR09b]. The most obvious application of voting systems might be political elections. There are also several applications in the fields of rank aggregation and multi-agent systems. Besides work that focuses on the problem to determine the winner or an optimal ranking for different voting systems, a significant number of papers also investigates how an external agent or a group of voters can influence the election in favor or disfavor of a distinguished candidate. Concrete scenarios of influencing are manipulation [BTT89, BFH<sup>+</sup>08, CSL07, MPRZ08], electoral control [BTT92, FHHR09a, HHR05], lobbying [CFRS07], and bribery [FHHR09a]. This shows that investigations in this field are of high interest. In this section we present a directed graph problem that is closely related to control in the Llull voting rule. We start with an introduction to this rule and introduce the corresponding graph problem.

**Llull voting.** Formally, an *election*  $(V, C)$  consists of a multiset of votes  $V$  and a set of candidates  $C$ . A *vote* is a preference list over all candidates. In an election, we either ask for a *winner*, that is, one of the candidates who are “best” in the election, or for a *unique winner*. Of course, a unique winner does not always exist. We only consider the unique-winner case for our control variant, but our results can be easily modified to work for the winner case as well.

The term Llull<sup>1</sup> voting was introduced by Faliszewski et al. [FHHR09a]. It is based on pairwise comparisons between candidates: A candidate wins the pairwise contest against another candidate if it beats the other candidate in more than half of the

---

<sup>1</sup>Llull is the special case of Copeland <sup>$\alpha$</sup>  where  $\alpha = 1$ .

votes. The winner of a pairwise contest gets one point and the loser receives no point. If two candidates are tied, both candidates get one point. A Llull winner is a candidate with the highest score. It is used for example in sport tournaments, chess, or in football leagues, where the teams or players can be considered as candidates. In the following we consider the concept “control of a voting rule”. To *control* an election, an external or internal agent, traditionally called the *chair*, is allowed or able to change the voting procedure to reach certain goals. For example, a typical question is how many candidates the chair has to delete to make his/her favorite candidate a unique winner. In most cases it is a desirable attribute of a voting rule to be either immune to control, that is, it is impossible to control the voting rule, or at least to be resistant to control, that is, the corresponding decision problem is NP-hard [BTT92]. Unfortunately, NP-hardness does only imply computational hardness in the worst case. There might be special inputs where it is easy to decide the corresponding problem.

Regarding the complexity of control, Llull voting is resistant to constructive candidate control and vulnerable for destructive candidate control [FHHR09a]. Thus, investigating the parameterized complexity of control of Llull voting helps us to extend our knowledge about the “danger of control of Llull election”. This is what we do in Chapter 4. Therefore, we introduce the directed graph problem which corresponds to candidate control in Llull elections.

**Minimum Indegree Deletion.** A Llull election can be depicted by a directed graph where the candidates are represented as vertices and there is an arc from vertex  $c$  to vertex  $d$  if and only if the corresponding candidate  $c$  defeats the corresponding candidate  $d$  in the pairwise comparison contest. Obviously, the Llull score of a candidate  $c$  can be considered as the total number of candidates minus the number of candidates that beat  $c$  in the pairwise comparison. Thus, a Llull winner corresponds to a vertex with minimum indegree. Of course, the deletion of a vertex corresponds to the deletion of a candidate in the election. These observations motivate the introduction of the following directed graph problem which was originally introduced in [BU09]:

MINIMUM INDEGREE DELETION

*Given:* A directed graph  $D = (W, A)$ , a distinguished vertex  $w_c \in W$ , and an integer  $k \geq 1$ .

*Question:* Is there a subset  $W' \subseteq W \setminus \{w_c\}$  of size at most  $k$  such that  $w_c$  is the only vertex that has minimum indegree in  $D - W'$ ?

The equivalence of MINIMUM INDEGREE DELETION to constructive control by deleting candidates in Llull elections was shown in [BU09].

## 2.2 Minimum Degree Deletion

Constructive control by deleting candidates in Llull elections leads to the directed MINIMUM INDEGREE DELETION. From the theoretical point of view, it is an intuitive task to investigate its undirected variant. From the practical point of view,

a corresponding voting problem can be formulated, too: Given is a *social network*, that is, an undirected graph  $G = (V, E)$  where vertices correspond to subjects and edges correspond to relations between two subjects. For example, consider the relation “disharmony between two subjects” and the subjects are candidates of a voting rule where the candidate with fewest disharmonies wins. Now, the chair is asked to control the election by removing a specified number of candidates from the network to make a specific candidate the winner of the election. The corresponding graph problem MINIMUM DEGREE DELETION is given in the following:

**MINIMUM DEGREE DELETION**

*Given:* An undirected graph  $G = (V, E)$ , a distinguished vertex  $w_c \in V$ , and an integer  $k \geq 1$ .

*Question:* Is there a subset  $V' \subseteq V \setminus \{w_c\}$  of size at most  $k$  such that  $w_c$  is the only vertex that has minimum degree in  $G - V'$ ?

Clearly, removing candidates from the social network corresponds to removing candidates in the graph  $G$ . The equivalence of the graph problem and constructive control of the voting rule by removing candidates is easy to see.

## 2.3 Bounded Degree Deletion

The first two problems ask for a vertex subset of a specific size whose removal satisfies the graph property “only a distinguished vertex has minimum (in)degree”. The problem which is introduced in this section no-longer has a distinguished candidate, but an upper bound on the degree. This means, the underlying graph property is “the maximum degree of the vertices in the graph is bounded by  $d$ ” for a specific positive integer  $d$  which is given in the input. BOUNDED DEGREE DELETION was introduced in [Mos10] and is formally defined as follows:

**BOUNDED DEGREE DELETION**

*Given:* An undirected graph  $G = (V, E)$ , and integers  $d \geq 0$  and  $k \geq 0$ .

*Question:* Does there exists a subset  $V' \subseteq V$  of size at most  $k$  whose removal from  $G$  yields a graph in which each vertex has degree at most  $d$ ?

Whereas MINIMUM INDEGREE DELETION and MINIMUM DEGREE DELETION correspond to problems in computational social choice, BOUNDED DEGREE DELETION has applications in computational biology: In the analysis of genetic networks based on micro array data, a central tool is to find cliques or “near-cliques” [BCK<sup>+</sup>05, CLS<sup>+</sup>05]. Searching for cliques is a computational hard problem. Here, VERTEX COVER as complementary dual problem to clique detection could be used very successful instead of direct clique detection. A mathematical concept for near-cliques is the concept of  $s$ -plexes, that are, subsets of vertices such that each  $s$ -plex vertex is connected to all other vertices in the  $s$ -plex but to  $s - 1$ . Note that cliques are 1-plexes. BOUNDED DEGREE DELETION is the complementary dual problem to  $s$ -plex detection [Mos10]. Since BOUNDED DEGREE DELETION is a vertex deletion problem for the hereditary graph property “each vertex has degree at most  $d$ ” [Mos10],

NP-completeness is given due to a general result by Lewis et al. [LY80]. It follows from the reduction in [LY80] that BOUNDED DEGREE DELETION cannot be approximated better than VERTEX COVER. Thus, an approximation lower bound of 1.36 assuming  $P \neq NP$  [DS05] as well as the APX-hardness of VERTEX COVER [PY91] can be carried over. The best known approximation factors are 2 [Fuj98] for  $d = 1$  and  $2 + \log(d)$  [OB03] for  $d \geq 2$ .

Finally we introduce some general terms which are used in the analysis of all three problems. A yes-instance of each problem can be detected through finding a vertex subset of bounded size whose removal satisfies the problem-specific graph property. We denote such subsets as *solution sets*. The result of removing a solution set from the input graph is called *solution graph*.

### 3 Feedback sets

As mentioned in the introduction, parameterized algorithms are designed to confine the combinatorial explosion to a parameter of the input instance. Popular parameters are, for example, the solution size for problems asking for a specific “solution set”. Besides very natural parameters such as “number of edges” in graph problems, “number of candidates” in voting problems or “size of the alphabet” in string-based problems, one is often interested in structural parameters like “average distance” or “size of a vertex cover” (see [Nie10] for a survey).

There is a huge class of graph problems which are easy to solve on acyclic graphs, but NP-hard in general. Later on, we will see that every graph problem based on a graph property expressible by some specific (non-trivial) logic is solvable in linear time on trees. The algorithms which follow directly from this result are quite impractical, but it is a good classification tool. In most cases, there are simple and direct polynomial-time algorithms for problems whose input graph is acyclic. In this way, a parameter that measures the “degree of acyclicity” of a graph is may be a good candidate for fixed-parameter tractability results. Often, one uses the treewidth and designs algorithms that operate on tree decompositions. In this work, we will also focus on two weaker parameters which will be introduced in this section.

There are three reasons for working also on other parameters that measure the “degree of acyclicity”, different from treewidth: The first reason is that computing an optimal tree decomposition (or even the treewidth) is difficult: The corresponding decision problem is NP-hard. Although there are some nice theoretical results and some work on practical computation of the treewidth of small graphs [Bod06, BGK08], no efficient algorithm that computes an optimal tree decomposition is known. Notably, it is an open question whether there is a constant-factor approximation for determining the treewidth of a graph [Bod08].

The second reason is that there is no simple explicit analogy to treewidth on directed graphs: A concept of “directed treewidth” was develop by Johnson et al. [JRST01]. Independently, the concept “DAG-width” was introduced in [BDHK06] as well as in [Obd06]. Alternatively, another concept “d-width” was introduced by Safari [Saf05]. Finally, a fourth concept “Kelly-width” was proposed by Hunter et al. [HK07]. Although there is a tendency to “Kelly-width” [MATV10], it is not even clear which of these concepts is the best analogy to undirected treewidth.

The third reason is that there are also problems that are not fixed-parameter tractable with respect to the parameter treewidth. Examples are CAPACITATED VERTEX COVER and CAPACITATED DOMINATING SET [DLSV08]. A weaker measure of the “degree of acyclicity” may lead to a parameter that allows for fixed-parameter tractability. To the best of our knowledge, there was no work with main focus on the parameterized complexity with respect to parameters that “measure

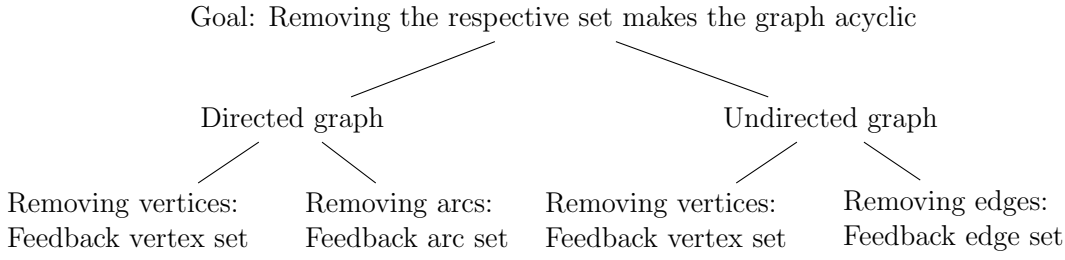


Figure 3.1: Overview: We consider four types of feedback sets. The corresponding parameters are the cardinalities of these sets.

the distance of the input graph to an acyclic graph” so far.

Informally speaking, a *feedback set* is a substructure of the graph such that its deletion makes the graph acyclic. Depending on the type of input (directed or undirected graph) and on the structural elements (vertices or edges/arcs), we consider four parameters (see Figure 3.1).

We start with the formal definition of the directed case: Let  $G = (V, A)$  be a directed graph. One calls a subset of vertices  $V' \subseteq V$  (*directed*) *feedback vertex set* if  $G - V'$  is acyclic. A subset of arcs  $A' \subseteq A$  is called *feedback arc set* if  $G' := (V, A \setminus A')$  is acyclic. Analogously, let  $G = (V, E)$  be an undirected graph. One calls a subset of vertices  $V' \subseteq V$  (*undirected*) *feedback vertex set* if  $G - V'$  is acyclic. A subset of edges  $E' \subseteq E$  is called *feedback edge set* if  $G' := (V, E \setminus E')$  is acyclic. Determining the sizes of these feedback sets leads directly to the following decision problems:

#### UNDIRECTED FEEDBACK VERTEX SET

*Given:* An undirected graph  $G = (V, E)$  and an integer  $k \geq 1$ .

*Question:* Is there an undirected feedback vertex set of size at most  $k$ ?

UNDIRECTED FEEDBACK VERTEX SET is NP-complete [Kar72]. With deterministic fixed-parameter algorithms it can be solved in  $O(5^k \cdot k \cdot n^2)$  time [CFL<sup>+</sup>08]. There is a randomized algorithm which solves UNDIRECTED FEEDBACK VERTEX SET in  $O(c \cdot 4^k \cdot kn)$  time by finding an undirected feedback vertex set of size  $k$  with probability at least  $1 - (1 - 4^{-k})^{c4^k}$  for an arbitrary constant  $c$ .

#### DIRECTED FEEDBACK VERTEX SET

*Given:* A directed graph  $G = (V, A)$  and an integer  $k \geq 1$ .

*Question:* Is there a directed feedback vertex set of size at most  $k$ ?

DIRECTED FEEDBACK VERTEX SET is also NP-complete. This follows directly from the reduction given by Karp in [Kar72]. However, recent studies proved its fixed-parameter tractability, showing that it can be solved in  $O(4^k \cdot k! \cdot n^4 \cdot k^3)$  time [CLL<sup>+</sup>08].

#### FEEDBACK ARC SET

*Given:* A directed graph  $G = (V, A)$  and an integer  $k \geq 1$ .

*Question:* Is there a feedback arc set of size at most  $k$ ?

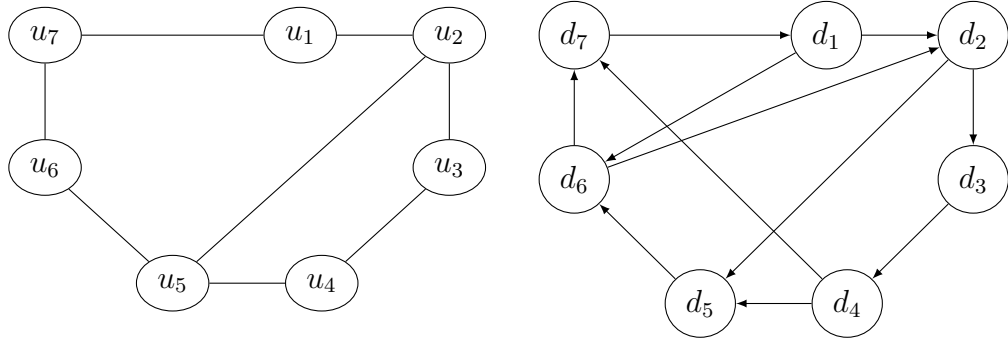


Figure 3.2: Feedback sets examples. Here we have an undirected and a directed graph. In the undirected graph there is a feedback vertex set of size one ( $\{u_2\}$ ) and a feedback edge set of size two (for example  $\{\{u_2, u_5\}, \{u_6, u_7\}\}$ ). In the directed graph there is a feedback vertex set of size two ( $\{d_1, d_2\}$ ) and a feedback arc set of size three (for example  $\{(d_2, d_3), (d_2, d_5), (d_1, d_6)\}$ ).

The directed FEEDBACK ARC SET is NP-complete [Kar72]. It was shown by Even et al. [ENSS98] that FEEDBACK ARC SET and DIRECTED FEEDBACK VERTEX SET can be reduced in linear time from one to each other retaining the parameter “size of a feedback set”. Hence, FEEDBACK ARC SET can also be solved in  $O(4^k \cdot k! \cdot n^4 \cdot k^3)$  time.

#### FEEDBACK EDGE SET

*Given:* A undirected graph  $G = (V, E)$  and an integer  $k \geq 1$ .

*Question:* Is there a feedback edge set of size at most  $k$ ?

The undirected FEEDBACK EDGE SET is polynomial-time solvable. It is easy to see that a (minimum) feedback edge set can be found by depth-first search. One just has to find a spanning tree. A minimal feedback edge set consists of the edges that are not in this tree.

In the following, we always talk about *feedback vertex sets* and the problem FEEDBACK VERTEX SET in both cases, the directed and the undirected, whenever it is clear which of them is considered. Furthermore, the parameters are called “size of a feedback vertex set” and so on. Of course, in most cases it is useful to know the size of the respective minimum feedback set. However, the algorithms do not depend on the minimality of the parameter value. This is an advantage when we use heuristics or approximation algorithms to find the feedback sets. There are cases where one has to know a concrete feedback set and the running time depends on the size of this set (see Section 5.4.1, Section 5.4.2, and Section 6.2). In contrast, there are also cases where the size of the feedback set is used indirectly to prove the worst-case running time (see Section 4.3, Section 5.2, and Section 5.3.2). There, it is not even necessary to know any feedback vertex set.

Some results can be carried over from one parameter to the other:

**Definition 3.** Let  $a$  and  $b$  denote two parameters. If  $a \leq b$  for each instance, then one says  $a$  is **stronger** than  $b$  (and  $b$  is **weaker** than  $a$ ).

Hardness results like  $W[t]$ -hardness for some positive integer  $t$  or non-existence of a problem kernel can be carried over from the weaker to the stronger parameter. In contrast, results like membership to a parameterized complexity class can be carried over from the stronger to the weaker parameter. It is easy to see that the treewidth of a graph is at most the size of a feedback vertex set of the same graph. Furthermore, for each feedback edge set respectively for each feedback arc set there is a feedback vertex set of at most the same size. Hence, “treewidth” is a stronger parameter than “size of a feedback vertex set” which is a stronger parameter than “size of a feedback edge set” respectively “size of a feedback arc set”.



## 4 Minimum Indegree Deletion

In this chapter, we analyze the parameterized complexity of MINIMUM INDEGREE DELETION. The problem is motivated as graph problem corresponding to constructive control by deleting candidates for Llull voting (see Chapter 2). The inputs are a directed graph  $D = (W, A)$ , a distinguished vertex  $w_c \in W$ , and an integer  $k \geq 1$ . The question is whether there is a subset  $W' \subseteq W \setminus \{w_c\}$  of size at most  $k$  such that  $w_c$  is the only vertex that has minimum indegree in  $D - W'$ . In most scenarios, computational hardness for control of a voting rule is a desirable attribute. However, NP-hardness should only be a first step in this regard, because the hardness does not necessarily hold for special cases of the input which can be typically in real-world instances. A parameterized analysis will help to discuss computational hardness for special types of input to the voting rule. We present intractability and tractability results for the parameters  $s_v$  := “size of a feedback vertex set”,  $s_a$  := “size of a feedback arc set”,  $k$  := “number of vertices to remove”, and  $d$  := “maximum degree of a vertex” and their combinations. Our results are summarized in Figure 4.1.

### 4.1 Known results

We briefly discuss previous result [BU09]: There is a simple polynomial-time algorithm that solves MINIMUM INDEGREE DELETION in acyclic directed graphs. In contrast, MINIMUM INDEGREE DELETION is W[2]-complete with respect to  $k$  := “size of a solution set”. This even holds if the input graph is a tournament.

Parameterized complexity:		
	single parameter	combined with $k$
$s_v$	<b>W[2]-hard, in XP</b>	<b>FPT</b>
$s_a$	<b>W[2]-hard, in XP</b>	<b>FPT</b>
$k$	W[2]-complete	
$d$	FPT	FPT

	parameter description
$s_v$	size of a feedback vertex set
$s_a$	size of a feedback arc set
$k$	size of a solution set
$d$	maximum degree of a vertex

Figure 4.1: Overview of the parameterized complexity of MINIMUM INDEGREE DELETION. New results are in boldface. The remaining results are obtained from [BU09].

With respect to the parameter  $d :=$  “maximum indegree of a vertex” MINIMUM INDEGREE DELETION is fixed-parameter tractable. Inspired by the result for acyclic graphs we study parameters measuring the “degree of acyclicity” of the input graph in the following.

## 4.2 Feedback vertex/arc set size as parameter

Here, we show that MINIMUM INDEGREE DELETION is  $W[2]$ -hard with respect to the parameter  $s_v :=$  “size of a feedback vertex set” and the parameter  $s_a :=$  “size of a feedback arc set”. To this end, we need the concept of “domination in an undirected graph”.

**Definition 4.** *Let  $G = (V, E)$  be an undirected graph. We say that a vertex  $d$  **dominates** a vertex  $v$  if  $d = v$  or  $d$  is a neighbor of  $v$ . A subset  $D \subseteq V$  is called **dominating set** if every vertex in  $V$  is dominated by a vertex in  $D$ .*

In the following, we provide parameterized reductions from DOMINATING SET, which is  $W[2]$ -complete with respect to the parameter  $k :=$  “size of the dominating set”.

DOMINATING SET

*Given:* An undirected graph  $G = (V, E)$  and an integer  $k \geq 1$ .

*Question:* Is there a dominating set of size at most  $k$ ?

**Parameterized reduction.** The following reduction is illustrated in Figure 4.2. Given a DOMINATING SET instance  $(G^* = (V^*, E^*), k)$  with  $V^* = \{v_1^*, v_2^*, \dots, v_n^*\}$ , we construct a directed graph  $G$  with feedback vertex set size  $k + 1$  and feedback arc set size  $(k + 1)^2$  such that  $(G, w_c, n - k)$  is a yes-instance of MINIMUM INDEGREE DELETION if and only if  $(G^*, k)$  is a yes-instance of DOMINATING SET. This means we have a parameterized reduction if we can do the construction in polynomial time. The vertex set of  $G$  consists of  $w_c$  and the union of the following disjoint vertex sets:

- $V, D$ , each containing one vertex for every vertex in  $V^*$ ,
  - $V := \{v_i \mid i \in \{1, \dots, n\}\}$  represents the set of “dominating vertices” which is illustrated by the nodes  $v_1, v_2$  and  $v_n$  in Figure 4.2,
  - $D := \{d_i \mid i \in \{1, \dots, n\}\}$  represents the set of “dominated vertices” which is illustrated by the nodes  $d_1, d_2$  and  $d_n$  in Figure 4.2.

The arcs between vertices in  $V$  and  $D$  ensure that “undominated vertices in a solution graph would have a lower indegree than  $w_c$ ”. They are illustrated by dotted arrows in Figure 4.2.

- $X, Y, Z$ , each containing  $k + 1$  vertices,
  - $X := \{x_i \mid i \in \{1, \dots, k + 1\}\}$ ,
  - $Y := \{y_i \mid i \in \{1, \dots, k + 1\}\}$ ,
  - $Z := \{z_i \mid i \in \{1, \dots, k + 1\}\}$ .



start set	end set
$\{x_1, \dots, x_k\}$	$\{s_{i,j} \mid i \in \{1, \dots, n\} \text{ and } j \in \{1, \dots, n\}\}$
$\{x_2, \dots, x_{k+1}\}$	$\{s_{i,j} \mid i \in \{1, \dots, n\} \text{ and } j \in \{n+1, \dots, 2n\}\}$
$\{y_1, \dots, y_k\}$ to	$\{s_{i,j} \mid i \in \{1, \dots, n\} \text{ and } j \in \{2n+1, \dots, 3n\}\}$
$\{y_2, \dots, y_{k+1}\}$	$\{s_{i,j} \mid i \in \{1, \dots, n\} \text{ and } j \in \{3n+1, \dots, 4n\}\}$
$\{z_1, \dots, z_k\}$	$\{s_{i,j} \mid i \in \{1, \dots, n\} \text{ and } j \in \{4n+1, \dots, 5n\}\}$
$\{z_2, \dots, z_{k+1}\}$	$\{s_{i,j} \mid i \in \{1, \dots, n\} \text{ and } j \in \{5n+1, \dots, 6n\}\}$

Figure 4.3: Concrete assignment of the arcs that are incident to vertices in  $\bigcup_{i=1}^n S_i$ . For each row of the table there is one arc from each vertex from the start set to each vertex from the end set.

These vertex sets are designed to “increase the indegree of vertices without increasing the size of a (minimum) feedback vertex set”. We will see that each of them is a feedback vertex set and every cycle in  $G$  passes through at least one vertex in  $X$ ,  $Y$ , and  $Z$ . The set  $X$  is illustrated by the nodes  $x_1$ ,  $x_2$ , and  $x_{k+1}$ ,  $Y$  is illustrated by  $y_1$ ,  $y_2$ , and  $y_{k+1}$ , and  $Z$  is illustrated by  $z_1$ ,  $z_2$ , and  $z_{k+1}$  in Figure 4.2.

- $S_1, \dots, S_n$  each containing  $6n$  vertices.

These vertex sets are designed to ensure that the set of removed vertices corresponding to any yes-instance  $(G, w_c, n-k)$  of MINIMUM INDEGREE DELETION only contains vertices of  $V$ . We will show that removing vertices that are not in  $V$  would be “punished by being forced to remove more than  $n$  further vertices in  $\bigcup_{i=1}^n S_i$ ”. Each set  $S_i$  is illustrated by the nodes  $s_{i,1}$ ,  $s_{i,2}$ , and  $s_{i,6n}$  in Figure 4.2.

The arcs of  $G$  are designed as follows:

- There is an arc from  $v_i$  to  $d_j$  if and only if  $v_i^*$  dominates  $v_j^*$ . These arcs are illustrated by dotted arrows in Figure 4.2.
- There is an arc from each vertex in  $V$  to  $w_c$ . These arcs are illustrated by thin arrows from the nodes  $v_1$ ,  $v_2$ , and  $v_n$  to the node  $w_c$  in Figure 4.2.
- There is an arc from each vertex in  $X$  to each vertex in  $Y$ , from each vertex in  $Y$  to each vertex in  $Z$ , and from each vertex in  $Z$  to each vertex in  $X$ . These arcs are illustrated by thin arrows between the nodes  $x_1$ ,  $x_2$ ,  $x_{k+1}$ ,  $y_1$ ,  $y_2$ ,  $y_{k+1}$ ,  $z_1$ ,  $z_2$ , and  $z_{k+1}$  in Figure 4.2.
- There are  $k$  arcs from arbitrary vertices in  $X \cup Y \cup Z$  to each vertex in  $D$  and  $k+1$  arcs from arbitrary vertices in  $X \cup Y \cup Z$  to each vertex in  $V$ . Outgoing arcs are illustrated by fat lines from the nodes  $x_1$ ,  $x_2$ ,  $x_{k+1}$ ,  $y_1$ ,  $y_2$ ,  $y_{k+1}$ ,  $z_1$ ,  $z_2$ , and  $z_{k+1}$  to the box B1. Ingoing arcs are illustrated by two fat lines from the box B1 to the boxes B2 and B3 and by fat arrows from the boxes B2 and B3 to the nodes  $v_1$ ,  $v_2$ ,  $v_n$ ,  $d_1$ ,  $d_2$ , and  $d_n$  in Figure 4.2.

- For each  $i \in \{1, \dots, n\}$  there is an arc from  $d_i$  and further  $k$  outgoing arc from vertices in  $X \cup Y \cup Z$  to each vertex in  $S_i$  such that every vertex in  $X \cup Y \cup Z$  has at least  $n$  outneighbors in  $\bigcup_{i=1}^n S_i$ . Removing a vertex in  $X \cup Y \cup Z$  decreases the indegree of its  $n$  outneighbors in  $\bigcup_{i=1}^n S_i$  too much. A concrete arc assignment is given in Figure 4.3. The outgoing arcs from vertices in  $D$  are illustrated by thin arrows from the nodes  $d_1, d_2$ , and  $d_n$  to the nodes  $s_{1,1}, s_{1,2}, s_{1,6n}, s_{2,1}, s_{2,2}, s_{2,6n}, s_{n,1}, s_{n,2}$ , and  $s_{n,6n}$ . The outgoing arcs from vertices in  $X \cup Y \cup Z$  are illustrated by a fat line from B1 to B4, a fat line from B4 to B5, and fat arrows from the box B5 to the nodes  $s_{i,1}, s_{i,2}$ , and  $s_{i,6n}$  for  $i \in \{1, \dots, n\}$  in Figure 4.2.

The construction ensures that an optimal solution deletes  $n - k$  vertices from  $V$  and no other vertex. This will be proved in detail later on, but it is necessary for the further argumentation to investigate the indegrees of the vertices: The indegree of  $w_c$  is  $n$  and the indegree of each vertex in  $V$  is  $k + 1$ . Since each vertex  $v_i^*$  dominates itself, the vertices in  $D$  have indegree at least  $k + 1$ . Every vertex in  $X \cup Y \cup Z$  has trivially also indegree  $k + 1$ . What remains are the vertex sets  $S_1, \dots, S_n$ . The vertices in  $\bigcup_{i=1}^n S_i$  have also indegree  $k + 1$ . In the following we give an intuition why they are useful: We want to ensure that if  $(G, w_c, n - k)$  is a yes-instance of MINIMUM INDEGREE DELETION, then every solution set only contains vertices of  $V$  and especially no vertices of  $D$ . For each vertex  $d_i$  with  $i \in \{1, \dots, n\}$  there is a set of vertices  $S_i := \{s_{i,1}, \dots, s_{i,6n}\}$  with an outgoing arc to each of the vertices in  $S_i$ . This realizes the “punishment”: Removing  $d_i$  is not possible without removing all vertices in  $S_i$  (which are more than  $n$ ). A similar argumentation holds for the vertices in  $X \cup Y \cup Z$ , because each of them has at least  $n$  outneighbors in  $\bigcup_{i=1}^n S_i$ . This finishes the description of the construction. Now, we give several lemmata and observations to prove the correctness of the construction, that is,  $(G^*, k)$  is a yes-instance of DOMINATING SET if and only if  $(G, w_c, n - k)$  is a yes-instance of MINIMUM INDEGREE DELETION.

**Lemma 1.** *If  $(G^*, k)$  is a yes-instance of DOMINATING SET, then  $(G, w_c, n - k)$  is a yes-instance of MINIMUM INDEGREE DELETION.*

*Proof.* Let  $(G^*, k)$  be a yes-instance and  $V_d^* \subseteq V^*$  be a dominating set of size  $k$  for  $G^*$ . Now, we delete a vertex  $v_i \in V$  from  $G$ , if  $v_i^* \notin V_d^*$ . Since  $V_d^*$  is of size  $k$  and  $V$  of size  $n$ , we deleted  $n - k$  vertices. The vertex  $w_c$  has now indegree  $k$ . We have to show that each other vertex has indegree at least  $k + 1$ . By construction every vertex in  $G$  has indegree at least  $k + 1$ . Since we removed only vertices in  $V$ , the only vertices which can have a decreased indegree are vertices in  $D$ . (These are the only vertices which can be outneighbors of a vertex in  $V$ .) Due to the fact, that  $V_d^*$  is a dominating set, every vertex in  $D$  keeps at least one inneighbor in  $V$ . By construction there are  $k$  further inneighbors in  $X \cup Y \cup Z$ . Hence, each vertex in  $D$  has at least indegree  $k + 1$ . Trivially, all other vertices keep indegree at least  $k + 1$ , because we did not remove any inneighbor. Thus,  $w_c$  has minimum indegree and  $(G, w_c, n - k)$  is a yes-instance of MINIMUM INDEGREE DELETION.  $\square$

Lemma 1 showed the direction from left to right of the parameterized equivalence between the DOMINATING SET solution and the MINIMUM INDEGREE DELETION

solution; now, we show the reverse direction. Consider  $(G, w_c, n - k)$  resulting from the parameterized reduction. Let  $M_d$  be a solution set for  $(G, w_c, n - k)$ . Several observations are very useful for the further argumentation.

**Observation 1.** *The constructed graph  $G$  has a feedback vertex set with at most  $k + 1$  vertices and a feedback arc set with at most  $(k + 1)^2$  arcs.*

*Proof.* We show by contradiction that  $G - X$  is acyclic. Assume that there is a (non-empty) cycle  $C = (c_1, \dots, c_l)$ . Since  $w_c$  and each vertex in  $\bigcup_{i=1}^n S_i$  has no outneighbor, neither  $w_c$  nor any vertex in  $\bigcup_{i=1}^n S_i$  can be part of  $C$ . Each vertex in  $d_i \in D$  has only outneighbors in  $S_i$ . Thus,  $C$  contains no vertex from  $D$ . Vertices from  $V$  have only outneighbors in  $\{w_c\} \cup D$  which implies that  $C$  does not contain any vertex from  $V$ . Vertices from  $Z$  have only outneighbors in  $\bigcup_{i=1}^n S_i \cup V \cup D$ , a set that contains no vertex in  $C$ . The remaining vertices from  $Y$  have only outneighbors in  $Z \cup V \cup D \cup \bigcup_{i=1}^n S_i$ . Thus,  $C$  must be empty; a contradiction.  $\square$

**Observation 2.** *It holds that  $w_c$  has indegree at least  $k$  in  $G - M_d$ .*

*Proof.* Assume that  $w_c$  has indegree at most  $k - 1$  in  $G - M_d$ . Since  $w_c$  has indegree  $n$  in the original graph  $G$ , we must have deleted  $n - k + 1$  inneighbors of  $w_c$ . Hence,  $M_d$  is a solution set of size at least  $n - k + 1$ ; a conflict.  $\square$

**Observation 3.** *The solution set  $M_d$  does not contain vertices of  $D$ ,  $X$ ,  $Y$ , or  $Z$ .*

*Proof.* Assume that  $u \in M_d$  is a vertex from  $D \cup X \cup Y \cup Z$ . By construction of  $G$ , vertex  $u$  has at least  $n$  outneighbors in  $\bigcup_{i=1}^n S_i$ . We call them  $S$ -outneighbors in the following. After removing  $u$ , every  $S$ -outneighbor must have indegree at most  $k$  in  $G - M_d$ , because it has indegree exactly  $k + 1$  in  $G$  (by construction). Since the final degree of  $w_c$  is at least  $k$  (see Observation 2) and  $w_c$  is the only vertex with minimum indegree in  $G - M_d$ , every  $S$ -outneighbor must be also in  $M_d$ . Thus,  $M_d$  has size at least  $n + 1$ ; a conflict.  $\square$

**Observation 4.** *In  $G - M_d$ , the distinguished vertex  $w_c$  has indegree exactly  $k$ .*

*Proof.* Assume that  $w_c$  has an indegree at least  $k + 1$ . Since  $w_c$  has to be the only vertex with minimum indegree, we have to delete each vertex with an indegree of at most  $k + 1$ . Remember that each vertex in  $X \cup Y \cup Z$  has an indegree of exactly  $k + 1$ . However, no vertex in  $X \cup Y \cup Z$  is part of the solution set (see Observation 3); a conflict.  $\square$

**Observation 5.** *It holds that  $M_d$  only contains vertices from  $V$ .*

*Proof.* In the original graph  $G$  the distinguished vertex  $w_c$  has indegree  $n$ . Due to Observation 4 the solution set  $M_d$  must contain at least  $n - k$  inneighbors of  $w_c$ . Due to the assumption that  $M_d$  has size at most  $n - k$  there is no other vertex in  $M_d$ .  $\square$

**Lemma 2.** *If  $(G, w_c, n - k)$  is a yes-instance of MINIMUM INDEGREE DELETION, then  $(G^*, k)$  is a yes-instance of DOMINATING SET.*

*Proof.* We show that if  $M_d$  is a solution set, then there is a dominating set  $V_d^* \subseteq V^*$  with at most  $k$  vertices. One can construct a dominating set  $V_d^*$  with  $|V_d^*| = k$  as follows: For each vertex  $v_i \in V \setminus M_d$  add vertex  $v_i^*$  to  $V_d^*$ . Due to Observations 4 and 5,  $V_d^*$  has size  $k$ . It remains to show that  $V_d^*$  is a domination set. Assume that there is a vertex  $v_f^*$  that is not dominated by any vertex in  $V_d^*$ . Thus, neither  $v_f$  nor any  $v_d \in N(v_f)$  is in  $M_d$ . Due to Observation 3, no vertex in  $D$  was deleted. Due to the construction of  $G$  there is no arc from any vertex in  $V$  to the undominated vertex  $d_f$ . Hence,  $d_f$  has indegree of  $k$  and  $w_c$  is not the only vertex with minimum indegree; a contradiction.  $\square$

Putting all together, we arrive at the following theorem:

**Theorem 1.** MINIMUM INDEGREE DELETION is  $W[2]$ -hard with respect to the parameter  $s_v$  as well as with respect to the parameter  $s_a$ .

*Proof.* We show that the transformation from  $(G^*, k)$  to  $(G, w_c, n - k)$  is a parameterized reduction: By construction, the transformation can be executed in  $f(k) \cdot \text{poly}(|x|)$  time with  $f(k)$  being a function only depending on  $k$  and  $|x|$  being the size of the input. Due to Observation 1 the parameter  $s_v$  ( $s_a$ ) is bounded by a function only depending on  $k$ . The equivalence follows Lemma 1 and Lemma 2.  $\square$

Theorem 1 provides a relative lower bound for the parameterized complexity with respect to the feedback set parameters  $s_v$  and  $s_a$ . An upper bound, namely the membership in XP, is a corollary of the main result of the next section.

### 4.3 Feedback vertex set size and solution size as combined parameter

Taking  $s_v :=$  “size of a feedback vertex set” ( $s_a :=$  “size of a feedback arc set”) as parameter MINIMUM INDEGREE DELETION does not yield fixed-parameter-tractability. As mentioned in Section 4.1,  $k :=$  “number of vertices to delete” as a parameter also leads to  $W[2]$ -hardness. So, it makes sense to consider the combined parameters  $(s_v, k)$  as well as  $(s_a, k)$  for MINIMUM INDEGREE DELETION. In this section, we show that each of these combined parameters leads to a fixed-parameter algorithm. The algorithm described in the following gets as input a directed graph, a distinguished vertex and a positive integer  $k$  and outputs a solution set of size  $k$ . Hence, with an additional factor  $k$  to the running time it also solves the corresponding minimization problem, where one asks for a minimum-size solution. The algorithm does not need to know or compute  $s_v$  or even a feedback vertex set.

We start with looking at the acyclic special case to describe the basic idea of the algorithm. One motivation to investigate MINIMUM INDEGREE DELETION with respect to the parameters that measure the “degree of acyclicity” is that the problem is NP-hard in general but polynomial-time solvable in the acyclic special case [BU09]. The corresponding algorithm is based on the fact that a directed acyclic graph always contains a vertex with indegree zero. It is an exhaustive application of the following step: If there is a vertex ( $\neq w_c$ ) with indegree zero, then remove it. This is trivially correct, since we want that  $w_c$  is the only vertex with minimum indegree (zero).

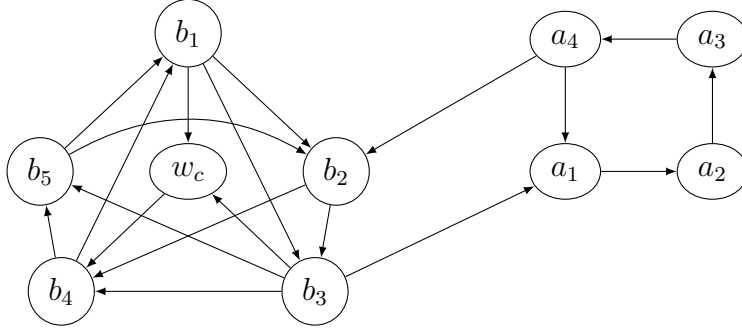


Figure 4.4: Directed cyclic graph where a minimum-size solution set of MINIMUM INDEGREE DELETION consists of  $b_1$  and  $b_3$  whereas  $a_2, a_3$ , and  $a_4$  have minimum indegree of 1.

There are two reasons why we cannot apply this algorithm to general directed graphs:

1. Adapting the idea directly by “removing every indegree-zero vertex except  $w_c$ ” is not correct, because it is possible that there is no vertex with indegree zero.
2. Adapting the idea in a more generalized way as “removing every vertex with minimum indegree except  $w_c$ ” is not correct. Unfortunately, it is even possible that we do not need to remove all or even any vertex that has minimum indegree in the input graph.

The first point is trivial. The second point is illustrated in Figure 4.4. As we see in this example another strategy to make  $w_c$  the only vertex with minimum indegree (besides removing vertices with smaller indegree) is to remove inneighbors of  $w_c$  to directly decrease its indegree. This leads to the algorithm **MinimumIndegreeDeletion** in Figure 4.5: To analyze the algorithm we take a closer look at our directed graph instance. First, we see that the minimum indegree in the solution graph is bounded by the size of any feedback vertex set.

**Lemma 3.** *Let  $G = (V, E)$  denote a directed graph. Let  $d_{min}$  denote the minimum indegree of the vertices in  $V$  and let  $V_f \subseteq V$  be a feedback vertex set. Then,  $|V_f| \geq d_{min}$ .*

*Proof.* There exists a vertex  $v_z$  with indegree zero in  $G - V_f$ , because  $G - V_f$  is acyclic. Since  $G$  has no vertex with indegree smaller than  $d_{min}$ , the feedback vertex set  $V_f$  must contain at least  $d_{min}$  inneighbors of  $v_z$ . Thus,  $|V_f| \geq d_{min}$ .  $\square$

A consequence is that the final indegree of  $w_c$  in a solution graph is at most the size of a feedback vertex set  $s_v$ . This holds due to Lemma 3 and the fact that a feedback vertex set in  $G$  implies a feedback vertex set in  $G - M_d$  of at most the same size. Hence, the loop in line 2 of the algorithm **MinimumIndegreeDeletion** is defined correctly. The main part (lines 4-14) of **MinimumIndegreeDeletion** is exhaustive exploration of the search space. It remains to show that the condition in line 3 is correct. Assume towards a contradiction that  $|N(w_c)| \geq k + i$  in a yes-instance of MINIMUM INDEGREE DELETION. Removing all but  $i$  neighbors implies a solution



```

1: procedure MINIMUMINDEGREEDELETION
2:   for each  $i := 0$  to  $s_v$  do
3:     if  $|N(w_c)| \leq i + k$  then
4:       for each size- $i$  subset  $U \subseteq N(w_c)$  do
5:         Remove  $D := N(w_c) \setminus U$  from  $G$ .
6:          $M_d := D$ 
7:         while there is a vertex  $d \neq w_c$  with indegree at most  $i$  do
8:           Remove  $d$  from  $G$ .
9:            $M_d := M_d \cup \{d\}$ 
10:        end while
11:        if  $|M_d| \leq k$  then
12:          return  $M_d$ 
13:        end if
14:      end for
15:    end if
16:  end for
17:  return “no”;
18: end procedure

```

Figure 4.5: Fixed-parameter algorithm that solves MINIMUM INDEGREE DELETION with respect to the parameter  $(s_v, k)$ . The variable  $i$  represents the final indegree of  $w_c$  in the solution graph.

set of size greater than  $k$ ; a contradiction. Hence, the algorithm is correct and we arrive at the following theorem:

**Theorem 2.** MINIMUM INDEGREE DELETION *with respect to the combined parameter  $(s_v, k)$ , with  $s_v$  being the size of a feedback vertex set and  $k$  being the size of a solution set, is solvable in  $O(s_v \cdot (k + 1)^{s_v} \cdot n^2)$  time.*

*Proof.* The correctness of the algorithm was already shown. It remains to analyze the running time. In the worst case, we have to start at most  $s_v$  times with the first (out-most) loop. In the second loop, we try at most  $\binom{s_v+k}{k}$  subsets. Thus, we have

$$\binom{s_v+k}{k} = \frac{(s_v+k)!}{k! \cdot (s_v+k-k)!} = \frac{\prod_{i=1}^{s_v} (k+i)}{s_v!} \leq \left(\frac{k+1}{1}\right)^{s_v}$$

subsets. The third loop can be done in  $O(n^2)$  time.  $\square$

Looking at the second loop of the algorithm helps us to see that MINIMUM INDEGREE DELETION with respect to the single parameter  $s_v$  is actually at least in XP. Branching into all possible subsets of  $i \leq s_v$  inneighbors of  $w_c$  is of course possible in  $O(n^i)$  time. In a more formal way, we arrive at the following corollary:

**Corollary 1.** *The problem MINIMUM INDEGREE DELETION with respect to the parameter  $s_v :=$  “size of a feedback vertex set” is in XP.*

*Proof.* Since  $(k+1)$ ,  $n$ , and  $s_v$  are upper bounded by the input size, this follows directly from Theorem 2. The running time of `MinimumIndegreeDeletion` is bounded by  $O(|x|^{s_v+3})$  with  $|x|$  being the input size.  $\square$



# 5 Minimum Degree Deletion

In this section, we investigate the parameterized complexity of MINIMUM DEGREE DELETION which can be considered as undirected variant of MINIMUM INDEGREE DELETION. It models electoral control by removing candidates of a special voting rule, see Section 2.2. In Chapter 4, we showed fixed-parameter intractability of MINIMUM INDEGREE DELETION with respect to the parameters  $s_v$  := “size of a feedback vertex set” and  $s_a$  := “size of a feedback arc set”, respectively. In contrast, we show in this chapter fixed-parameter tractability even for the stronger parameter “treewidth of the input graph”. Furthermore, we differentiate the parameters by comparing their kernel sizes. Firstly, we show that MINIMUM DEGREE DELETION is as well as its directed variant MINIMUM INDEGREE DELETION fixed-parameter intractable with respect to the parameter  $k$  := “number of vertices to delete”. An overview about the kernel sizes and the parameterized complexity with respect to the (combined) parameters is given in Figure 5.1.

Parameterized complexity:		
	parameter description	complexity
$t_w$	treewidth of the input graph	<b>FPT</b>
$s_v$	size of a feedback vertex set	<b>FPT</b>
$s_a$	size of a feedback arc set	<b>FPT</b>
$k$	size of a solution set	<b>W[1]-hard, in XP</b>
$d$	maximum degree of a vertex	FPT

Kernel sizes:		
	single parameter	combined with $k$
$t_w$	<b>no polynomial</b>	<b>no polynomial</b>
$s_v$	<b>no polynomial</b>	<b>no polynomial</b>
$s_a$	<b>vertex-linear</b>	<b>vertex-linear</b>
$k$	<b>no kernel</b>	
$d$	open	open

Figure 5.1: Overview of the parameterized complexity of MINIMUM DEGREE DELETION and the corresponding kernel sizes. New results are in boldface. The results for the parameter  $d$  := “maximum degree of a vertex” can be directly transferred from the results for the directed variant in [BU09]. A vertex-linear kernel is a problem kernel whose size is linear in the number of vertices.

## 5.1 Solution size as parameter

In this section, we analyze the parameterized complexity of MINIMUM DEGREE DELETION with respect to the parameter  $k :=$  “number of vertices to delete”. Using methods similar to those in Section 4.2 we show  $W[1]$ -hardness by presenting a parameterized reduction from INDEPENDENT SET with respect to the parameter “independent set size”. Below, we give a description and illustration of the reduction. We first show that one can assume that each INDEPENDENT SET instance has an even number of edges. Let  $G = (V, E)$  be an undirected graph with  $n := |V|$  and  $m := |E|$ . One can transform each instance with an odd value of  $m$  to a new instance with an even value of  $m$  such that the new instance is a yes-instance if and only if the original instance is a yes-instance and the parameter value does not change. To this end, we have to consider two cases: The first case is that the number of vertices is odd. Then we only have to add a new vertex and connect it to each vertex of the original graph. The second case is that the number of vertices is even. Here, we have to add a clique of three new vertices and connect each of these vertices of each vertex of the original graph. So, we get  $3n + 3$  new edges which is odd since  $n$  is even. Trivially, none of the new vertices will ever be a part of an independent set of size at least two<sup>1</sup> in both cases.

**Parameterized reduction.** The following reduction is illustrated in Figure 5.2. Given an INDEPENDENT SET instance  $(G^* = (V^*, E^*), k)$ , with  $V^* = \{v_1^*, v_2^*, \dots, v_n^*\}$  and  $E^* = \{e_1^*, e_2^*, \dots, e_m^*\}$ , we construct an undirected graph  $G$ , with a distinguished vertex  $w_c$  such that  $(G, w_c, k)$  is a yes-instance of MINIMUM DEGREE DELETION if and only if  $(G^*, k)$  is a yes-instance of INDEPENDENT SET. The vertex set of  $G$  consists of  $w_c$  and the union of the following disjoint vertex sets:

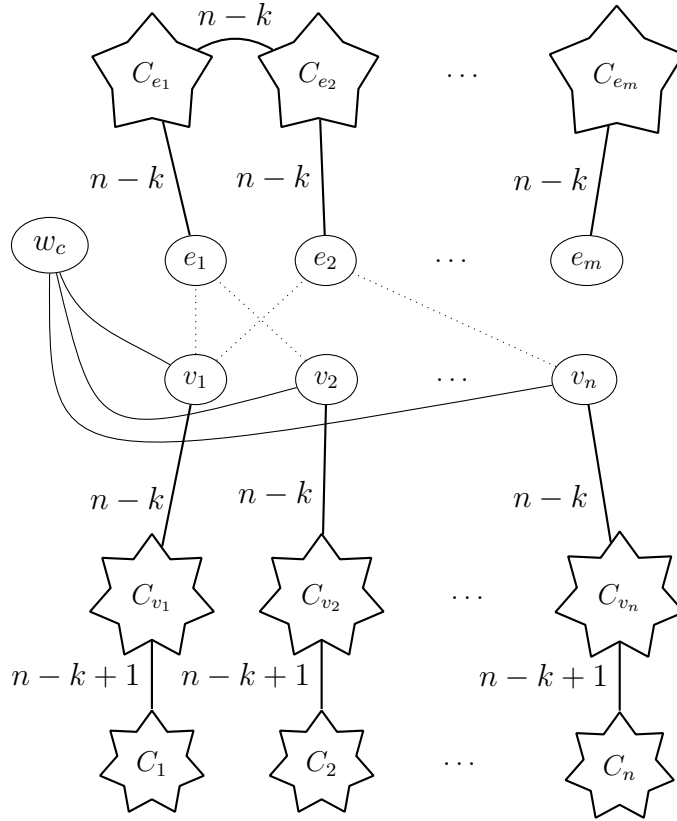
- $V$  containing one vertex for every vertex in  $V^*$  and  $E$  containing one vertex for every edge in  $E^*$ .
  - $V := \{v_i \mid i \in \{1, \dots, n\}\}$  represents the set of vertices which is illustrated by the nodes  $v_1, v_2$  and  $v_n$  in Figure 5.2.
  - $E := \{e_i \mid i \in \{1, \dots, m\}\}$  represents the set of “connections between vertices” which is illustrated by the nodes  $e_1, e_2$ , and  $e_m$  in Figure 5.2.
- For each vertex  $v_i \in V$  there are two cliques  $C_{v_i}$  and  $C_i$  of size  $n - k + 1$ , and for each vertex  $e_i \in E$  there is one clique  $C_{e_i}$  of size  $n - k$ . These cliques are illustrated by stars in Figure 5.2.

The edges (besides inner-clique edges) of  $G$  are drawn as follows:

- There is an edge between  $v_i$  and  $e_j$  if and only if  $v_i^*$  is incident to  $e_j^*$ . These edges are illustrated by dotted lines in Figure 5.2.
- The distinguished vertex  $w_c$  is connected to each vertex in  $V$ . The corresponding edges are illustrated by thin lines between the node  $w_c$  and the nodes  $v_1, v_2$ , and  $v_n$  in Figure 5.2.

<sup>1</sup>Every graph with at least one vertex has an independent set of size one.

Each vertex from  $C_{e_j}$  with  $j \in \{1, 3, 5, \dots, m-1\}$  is connected to exactly one vertex from  $C_{e_{j+1}}$ .



Exactly  $n - k$  vertices in  $C_x$  with  $x \in \{v_1, \dots, v_n\} \cup \{e_1, \dots, e_m\}$  are connected to the vertex  $x$ .

There is an edge between  $e_x$  and  $v_y$  if and only if  $v_y^*$  is incident to  $e_x^*$ .

Each vertex from  $C_{v_i}$  with  $i \in \{1, 2, \dots, n\}$  is connected to exactly one vertex from  $C_i$ .

Figure 5.2: MINIMUM DEGREE DELETION instance obtained from a parameterized reduction from an INDEPENDENT SET with an even number of edges. Each star  $C_x$  with  $x \in \{v_1, \dots, v_n\}$  represents a clique with  $n - k$  vertices. Each star  $C_x$  with  $x \in \{1, \dots, n\} \cup \{e_1, \dots, e_m\}$  represents a clique with  $n - k + 1$  vertices.

- Each vertex of every clique  $C_{e_j}$  with  $j \in \{1, 3, \dots, m-1\}$  is connected to exactly one (different) vertex in  $C_{e_{j+1}}$ . (Note that  $m-1$  is odd.) The corresponding edges are illustrated by a fat line between  $C_{e_1}$  and  $C_{e_2}$  which is labeled with  $n-k$  in Figure 5.2.
- Exactly  $n-k$  vertices of each clique  $C_x$  with  $x \in \{v_1, \dots, v_n, e_1, \dots, e_m\}$  are connected to the vertex  $x$ . The corresponding edges are illustrated by the remaining fat lines which are labeled with  $n-k$  in Figure 5.2.
- Each vertex of every clique  $C_{v_i}$  with  $i \in \{1, \dots, n\}$  is connected to exactly one (different) vertex in  $C_i$ . The corresponding edges are illustrated by fat lines which are labeled with  $n-k+1$  in Figure 5.2.

Our goal is to ensure that an optimal solution deletes  $k$  vertices from  $V$  and no other vertex. Therefore, we set the degree of  $w_c$  to  $n$  and for each other vertex to at least  $n-k+1$ . Each vertex in  $C_i$  with  $i \in \{1, \dots, n\}$  and  $C_{e_j}$  with  $j \in \{1, \dots, m\}$  has degree  $n-k+1$  and each vertex in  $C_{v_i}$  with  $i \in \{1, \dots, n\}$  has degree  $n-k+1$  or  $n-k+2$ . Each vertex  $e_j$  with  $j \in \{1, \dots, m\}$  has degree  $n-k+2$  and each vertex  $v_i$  with  $i \in \{1, \dots, n\}$  has degree at least  $n-k+1$ . This finishes the description of the construction. Now, we give several lemmata and observations to prove the correctness of the construction, that is,  $(G^*, k)$  is a yes-instance of INDEPENDENT SET if and only if  $(G, w_c, k)$  is a yes-instance of MINIMUM DEGREE DELETION.

**Lemma 4.** *If  $(G^*, k)$  is a yes-instance of INDEPENDENT SET, then  $(G, w_c, k)$  is a yes-instance of MINIMUM DEGREE DELETION.*

*Proof.* Let  $(G^*, k)$  be a yes-instance and  $V_d^* \subseteq V^*$  a size- $k$  set of independent vertices. We delete each vertex  $v_i \in V$  from  $G$ , if  $v_i^* \in V_d^*$ . Since  $|V_d^*| = k$  and  $|V| = n$ , we deleted  $k$  vertices and the vertex  $w_c$  has now degree of  $n-k$ . It holds that each vertex  $e_j$  with  $j \in \{1, \dots, m\}$  has degree at least  $n-k+1$ , because  $V_d^*$  is an independent set which means that at most one of the neighbors from  $e_j$  in  $V$  is deleted. Trivially, the degree of all other vertices is at least  $n-k+1$ . Thus,  $w_c$  has minimum degree and  $(G, w_c, k) \in \text{MINIMUM DEGREE DELETION}$ .  $\square$

Next, we will show the opposite direction of the equivalence. To this end, several observations are very useful for further argumentation. In the following, let  $G^* = (V^*, E^*)$  be an undirected graph and  $k$  a positive integer. We apply our reduction and denote the resulting graph and its vertices and vertex sets as described above. Let  $M_d$  denote a set of  $k$  vertices of  $G$  such that its deletion makes  $w_c$  become the only vertex with minimum degree.

**Observation 6.** *It holds that  $w_c$  has degree exactly  $n-k$  after deleting  $M_d$  from  $G$ .*

*Proof.* Assume towards a contradiction that the degree of  $w_c$  does not equal  $n-k$ . First consider the case that  $w_c$  has degree less than  $n-k$  after deleting  $M_d$  from  $G$ . In the original graph  $w_c$  has degree  $n$ . To reach degree less than  $n-k$  we have to delete more than  $k$  vertices; a contradiction. Second consider the case that  $w_c$  has degree more than  $n-k$  after deleting  $M_d$  from  $G$ . This means that the degree of  $w_c$  is at least  $n-k+1$ . Since  $M_d$  is a solution set, no other vertex with degree  $n-k+1$

can exist after deleting  $M_d$  from  $G$ . So, we have to delete more than  $n > k$  vertices, since there are more than  $n$  cliques with vertices with degree exactly  $n - k + 1$ ; a contradiction. Altogether,  $w_c$  has degree exactly  $n - k$  after deleting  $M_d$  from  $G$ .  $\square$

**Observation 7.** *It holds that  $M_d$  only contains vertices of  $v_i$  with  $i \in \{1, \dots, n\}$ .*

*Proof.* The argumentation is very simple. Due to Observation 6 it is clear that  $M_d$  contains at least  $k$  vertices  $v_i$  with  $i \in \{1, \dots, n\}$ . Since  $M_d$  has size  $k$ , there is no other vertex in  $M_d$ .  $\square$

**Lemma 5.** *If  $(G, w_c, k)$  is a yes-instance of MINIMUM DEGREE DELETION, then  $(G^*, k)$  is a yes-instance of INDEPENDENT SET.*

*Proof.* Let  $(G, w_c, k)$  be a yes-instance of MINIMUM DEGREE DELETION. We can build a size- $k$  independent set  $V_d^*$  as follows: For each vertex  $v_i \in M_d$  add vertex  $v_i^*$  to  $V_d^*$ . Due to Observations 6 and 7,  $V_d^*$  has size  $k$ . It remains to be shown that  $V_d^*$  is an independent set. Assume that there is an edge  $e_j^*$  with  $j \in \{1, \dots, m\}$  that is incident to any two vertices  $v_a, v_b \in V_d^*$ . Thus, both  $v_a$  and  $v_b$  are in  $M_d$ . Due to the construction of  $G$  the vertex  $e_j$  has degree of  $n - k$  after deleting  $v_a$  and  $v_b$  and must be deleted, too; a contradiction to Observation 7.  $\square$

Altogether, we arrive at the following:

**Theorem 3.** *MINIMUM DEGREE DELETION is  $W[1]$ -hard with respect to the parameter  $k := \text{“number of vertices to delete”}$ .*

*Proof.* We show that the transformation from  $(G^*, k)$  to  $(G, w_c, k)$  is a parameterized reduction: By construction, it can be executed in  $f(k) \cdot \text{poly}(|x|, k)$  time. The new parameter equals the original one. The equivalence follows Lemma 4 and Lemma 5.  $\square$

Theorem 3 provides a relative lower bound for the parameterized complexity with respect to the parameter  $k$ . An upper bound, namely the membership in XP, is quite easy to see. Simply checking for each  $V' \subseteq V$  whether  $w_c$  is the only vertex with minimum degree in  $G - V'$  already leads to the following:

**Proposition 1.** *MINIMUM DEGREE DELETION is in XP with respect to the parameter  $k := \text{“number of vertices to delete”}$ .*

## 5.2 Size of a feedback edge set as parameter

In Section 4.3, we showed that MINIMUM INDEGREE DELETION is fixed-parameter-tractable with respect to the combined parameter “size of a feedback vertex/arc set” and “size of a solution set”. In contrast, it is  $W[2]$ -hard for both single parameters (Section 4.2 and [BU09]). It is easy to adapt the algorithm of Section 4.3 to the undirected version MINIMUM DEGREE DELETION. Hence, we can show fixed-parameter tractability for the combined parameter “size of a feedback vertex/edge set” and “size of a solution set”. However, the parameterized intractability for the

single parameters “size of a feedback vertex set” and “size of a feedback arc/edge set” cannot be transferred as easily.

In contrast to the hardness results of the directed problem we will show fixed-parameter tractability for both parameters. We will start with a data reduction rule that achieves a *vertex-linear kernel*, that is, a problem kernel whose size is linear in the number of vertices, for MINIMUM INDEGREE DELETION with respect to the parameter “size of a feedback edge set”.

**A vertex-linear kernel.** Our kernelization is based on a simple data reduction rule. The following lemma ensures polynomial running time for this rule.

**Lemma 6.** *Let  $G = (V, E)$  be an undirected graph and  $k$  a positive integer. In  $O(n^2 \cdot k)$  time and  $O(n^2 + n)$  space one can determine whether there is a set of vertices  $M_d \subseteq V$  with  $|M_d| \leq k$  such that  $w_c$  is the only vertex with minimum degree in  $G - M_d$  and  $\deg(w_c) \leq 1$ .*

*Proof.* Determining whether there is a solution set  $M_d \subseteq V$  with  $|M_d| \leq k$  such that  $w_c$  is the only vertex with minimum degree and  $\deg(w_c) \leq 1$  works as follows: To manage the degree information of the vertices we use an adjacency matrix and store the sums of each columns and each row. The column and row sums give us directly the degree of each vertex. The initialization of the matrix costs  $O(n^2)$  time and  $O(n^2 + n)$  space. In a first step we remove all neighbors of  $w_c$  if  $\deg(w_c) = 0$  and all but one neighbor if  $\deg(w_c) = 1$ . Subsequently, we remove all vertices with degree at most  $\deg(w_c)$ . Removing one vertex costs  $O(n)$  time, because we need to update the matrix. There are at most  $k$  removal steps. If  $\deg(w_c) = 1$  there are up to  $n$  possible neighbor which are not removed. This means an additional factor of  $O(n)$  in the case  $\deg(w_c) = 1$ . Thus, we need  $O(n \cdot (k \cdot n)) = O(n^2 \cdot k)$  time and  $O(n^2 + n)$  space.  $\square$

### Reduction Rule “Remove Low Degree”

Let  $G = (V, E)$  be an undirected graph and  $k$  be a positive integer. We denote by  $\text{RLD}(G)$  the graph resulting by the following data reduction: If there is set of vertices  $M_d \subseteq V$  with  $|M_d| \leq k$  such that  $w_c$  is the only vertex with minimum degree and  $\deg(w_c) \leq 1$  in  $G - M_d$ , then replace  $G$  with a new graph which only contains the single vertex  $w_c$  and set the parameter to zero. Otherwise,  $w_c$  has degree at least two in every optimal solution, iteratively remove each vertex with degree at most two and decrease the parameter by one in each removal step.

Due to Lemma 6, Reduction Rule “Remove Low Degree” can be executed in polynomial time. It is easy to verify that the rule is sound. The next observation follows directly from the construction of  $\text{RLD}(G)$ .

**Observation 8.** *Every vertex ( $\neq w_c$ ) in  $\text{RLD}(G)$  has degree at least three.*

Now, we are ready to bound the number of vertices in  $\text{RLD}(G)$  with the help of this observation. For the ease of argumentation we firstly define a transformation which, given a forest, removes every inner vertex with degree two by connecting its neighbors.



**Definition 5.** Let  $G$  denote a forest. The function  $\text{DisLined}(G)$  denotes the result of the exhaustive application of the following procedure: If there is an induced  $P_3$  (path of length three) such that the middle vertex  $v'$  in  $P_3$  has degree two in  $G$ , then remove  $v'$  and connect both its neighbors.

**Theorem 4.** There is a  $2s_e$ -vertex kernel for MINIMUM DEGREE DELETION with respect to the parameter  $s_e := \text{“size of a feedback edge set”}$ . It is computable in  $O(n^2 \cdot k)$  time with  $n$  being the number of vertices and  $k$  being the number of vertices to delete.

*Proof.* Let  $G$  denote an undirected graph. We show that there are at most  $2 \cdot s_e$  vertices with  $s_e$  being the size of a feedback edge set in  $\text{RLD}(G)$ . Let  $E_d$  denote a feedback edge set of size  $s_e$ . Clearly,  $G - E_d$  is a forest. Since each vertex in  $G$  has degree at least three (Observation 8), each leaf in  $G - E_d$  must be incident to at least two edges in  $E_d$ . It holds that  $G - E_d$  contains  $l \leq s_e$  leaves, because each leaf must be incident to two edges of the feedback edge set and each edge of the feedback edge set can be incident to at most two leaves. Furthermore, the sum of incidences of the edges in  $E_d$  is  $2s_e$ . Each inner vertex of degree two in  $G - E_d$  must be incident to at least one edge in  $E_d$ . Since there are  $l$  leaves in  $G - E_d$ , only  $2s_e - 2l$  incidences are left over. Hence,  $G - E_d$  contains at most  $2s_e - 2l$  inner vertices with degree two. It holds that  $G - E_d$  has at most  $l$  inner vertices with degree at least three if  $\text{DisLined}(G - E_d)$  has at most  $l$  inner vertices with degree at least three. Even if  $\text{DisLined}(G - E_d)$  is a complete binary tree there are at most  $l/2 + l/4 + \dots + 1 = l$  inner vertices. Altogether,  $G$  has at most  $l + 2s_e - 2l + l = 2s_e$  vertices. The running time follows Lemma 6.  $\square$

**A search tree algorithm.** Of course, the  $2s_e$ -vertex-kernel already provides fixed-parameter tractability. The running time of a corresponding brute-force algorithm is  $O(4^{s_e} \cdot n^2)$  with  $s_e$  being the size of a feedback edge set. The polynomial factor is for checking the correctness of the “guessed” solution set. A simple refinement helps to give the search tree algorithm **MDD-search** (see Figure 5.3). The input is an instance of MINIMUM DEGREE DELETION  $(G = (V, E), w_c, k)$  and a feedback edge set  $E_f$  with  $|E_f| = s_e$ . We start with showing the correctness of **MDD-search**. After applying the data reduction rule (see line 2), **MDD-search** branches over each  $N' \subseteq N_E$  with  $N_E$  being the set of  $w_c$ -neighbors that are connected to  $w_c$  by an edge of  $E_d$  (see line 3). Here, the set  $N'$  expresses the “vertices from  $N_E$  that are part of the solution set”. **MDD-search** removes  $N'$  from the graph, decreases the parameter value, and applies the data reduction rule again (see line 4-6). Then, **MDD-search** branches over  $N'' \subseteq N(w_c) \setminus N_E$  which are the “remaining neighbors of  $w_c$  that are part of the solution set” (see line 7). The algorithm removes  $N''$  from the graph and decreases the parameter value (see lines 8-9). Finally, **MDD-search** determines the remaining part of the solution set by iteratively removing all vertices with degree  $\leq \deg(w_c)$  (see lines 10-13). Clearly, the algorithm is correct, because it finally branches over all neighbors of  $w_c$  that can be part of a solution set (see lines 2 and 9) and detects the remaining solution set vertices which are uniquely determined in each branching (see lines 10-13).

It remains to analyze the running time. The first loop (line 3) iterates  $O(2^{|N_E|})$  times. By definition,  $|N_E| \leq s_e$ . Now consider the graph  $G' - N_E$  in line 7. Clearly,

---

```

1: procedure MDD-SEARCH( $G = (V, E), w_c, k, E_f$ )
2:    $(G', k') := \text{RLD}(G)$ 
3:   for each  $N' \subseteq N_E$  with  $N_E := \{x \mid \{x, w_c\} \in E_f\}$  do
4:     Remove  $N'$  from  $G'$ .
5:      $k' := k - |N'|$ 
6:      $(G', k') := \text{RLD}(G')$ 
7:     for each  $N'' \subseteq (N(w_c) \setminus N_E)$  in  $G'$  do
8:       Remove  $N''$  from  $G'$ .
9:        $k' := k' - |N''|$ 
10:      while there is a vertex  $v$  with  $\deg(v) \leq \deg(w_c)$  in  $G'$  do
11:        Remove  $v$  from  $G'$ .
12:         $k' := k' - 1$ 
13:      end while
14:      if  $k' \geq 0$  then
15:        return “yes”
16:      end if
17:    end for
18:  end for
19:  return “no”
20: end procedure

```

Figure 5.3: Fixed-parameter algorithm that solves MINIMUM DEGREE DELETION with respect to the parameter in  $O(2^{s_e} \cdot n^2)$  time.

$G' - N_E$  has a feedback edge set  $E'_f$  with  $|E'_f| = s_e - |N_E|$ . Due to the proof of Theorem 4 we already know that  $(G' - N_E) - E'_f$  has at most  $|E'_f|$  leaves. Since the “remaining neighborhood of  $w_c$  in  $G'$ ”, namely  $N(w_c) \setminus N_E$  in  $G'$ , does not change after removing  $N_E$  and  $E'_f$  from  $G'$ ,  $|N(w_c) \setminus N_E|$  is also bounded by  $|E'_f|$ . Hence, the second loop iterates at most  $O(2^{|E'_f|}) = O(2^{s_e - |N_E|})$  times. The remaining operations can be done in  $O(n^3)$  time. Putting all together we arrive at the following:

**Theorem 5.** MINIMUM DEGREE DELETION is solvable in  $O(2^{s_e} \cdot n^3)$  time with  $s_e$  being the size of a feedback edge set and  $n$  being the number of vertices.

As already mentioned we also show fixed-parameter tractability for the parameter  $s_v$ . Since  $t_w :=$  “treewidth of the input graph” is a lower bound for  $s_v$ , fixed-parameter tractability for  $t_w$  would trivially imply fixed-parameter tractability for  $s_v$ . Hence, we start with the investigation of the parameterized complexity of MINIMUM DEGREE DELETION with respect to the parameter  $t_w$  in the following section.

### 5.3 Treewidth as parameter

In the previous sections, we have presented explicit fixed-parameter algorithms to prove that the problems are tractable with respect to the corresponding parameters. It is sometimes hard to find explicit algorithms that solve a parameterized problem. Fortunately, there are results that state that large classes of problems

can be solved in linear time when a tree decomposition with constant treewidth is known (see [Cou90, Cou09]). Note that the method stated in this section is of purely theoretical interest. The corresponding running times have huge constant factors and combinatorial explosions with respect to the parameter treewidth. Hence, for practical applications one should search for an effective problem-specific algorithm.

### 5.3.1 Monadic second-order logic

We use a tool called *monadic second-order logic* (or short *MSO*). This is an extension to the well-known first-order logic by quantification over sets. Courcelle's Theorem [Cou90] says that the verification of a graph property is fixed-parameter tractable with respect to the parameter treewidth if the property can be expressed with monadic second-order logic. An extensive overview about the field of monadic second-order logic can be found in [Cou09]. In the following, we describe the language and syntax of MSO-formulae. An MSO-formula consists of:

- an infinite supply of *individual variables*, by convention denoted by small letters  $x, y, z, \dots$ ,
- an infinite supply of *set variables*, by convention denoted by capital letters  $X, Y, Z, \dots$ ,
- to express graph properties two unary relations, by convention denoted as  $V$  and  $E$ , and a binary relation, by convention denoted as  $I$ , where
  - the relation  $V$  can be interpreted as “being a vertex”,
  - the relation  $E$  can be interpreted as “being an edge”,
  - and the relation  $I$  can be interpreted as “being incident”,
- some logical operators ( $\neg, \wedge, \vee, \rightarrow$ , and  $\leftrightarrow$ ) and the quantifiers  $\exists$  and  $\forall$ .

The relations will be used in prefix notation. For a graph  $G = (V, E)$  let  $U := V \cup E$ . An *assignment*  $\alpha$  for an MSO-formula maps each individual variable to an element of  $U$  and each set variable to a subset of  $U$ . One defines the concept of an assignment  $\alpha$  *satisfying* an MSO-formula  $\phi$ , written  $(G, \alpha) \models \phi$  for a given graph  $G$ . Now, we can define the *atomic* MSO-formulae and their semantics. Let  $G = (V, E)$  be a graph,  $x$  and  $y$  being individual variables, and  $X$  be a set variable. We have the following atomic MSO-formulae:

atomic formula	semantics
$x = y$	$(G, \alpha) \models x = y \Leftrightarrow \alpha(x) = \alpha(y)$
$Vx$	$(G, \alpha) \models Vx \Leftrightarrow \alpha(x) \in V$
$Ex$	$(G, \alpha) \models Ex \Leftrightarrow \alpha(x) \in E$
$Ixy$	$(G, \alpha) \models Ixy \Leftrightarrow \alpha(x) \in V \text{ is incident to } \alpha(y) \in E$
$Xx$	$(G, \alpha) \models Xx \Leftrightarrow \alpha(x) \in X$

Moreover, all other (more complex) MSO-formulae can be inductively built as follows:

- If  $\phi$  is an MSO-formula, then  $\neg\phi$  is an MSO-formula as well.

- If  $\phi$  and  $\psi$  are MSO-formulae, then  $\phi \wedge \psi$ ,  $\phi \vee \psi$ ,  $\phi \rightarrow \psi$ , and  $\phi \leftrightarrow \psi$  are MSO-formulae as well.
- If  $\phi$  is an MSO-formula,  $x$  is an individual variable, and  $X$  is a set variable, then  $\exists x\phi$ ,  $\forall x\phi$ ,  $\exists X\phi$ , and  $\forall X\phi$  are MSO-formulae as well.

Although “ $\rightarrow$ ” and “ $\leftrightarrow$ ” are not explicitly necessary we list them for sake of completeness. Their semantics is analogous to first-order logic by combining “ $\wedge$ ”, “ $\vee$ ”, and “ $\neg$ ”. The constructions have the following semantics:

construct	semantics
$\neg\phi$	$(G, \alpha) \models \neg\phi \Leftrightarrow (G, \alpha) \not\models \phi$
$\phi \wedge \psi$	$(G, \alpha) \models \phi \wedge \psi \Leftrightarrow (G, \alpha) \models \phi$ and $(G, \alpha) \models \psi$
$\phi \vee \psi$	$(G, \alpha) \models \phi \vee \psi \Leftrightarrow (G, \alpha) \models \phi$ or $(G, \alpha) \models \psi$
$\exists x$	$(G, \alpha) \models \exists x\phi \Leftrightarrow$ there exists an $a \in U$ such that $(G, \alpha_x^a) \models \phi$
$\forall x$	$(G, \alpha) \models \forall x\phi \Leftrightarrow$ for all $a \in U$ it holds that $(G, \alpha_x^a) \models \phi$
$\exists X$	$(G, \alpha) \models \exists X\phi \Leftrightarrow$ there exists an $A \subseteq U$ such that $(G, \alpha_X^A) \models \phi$
$\forall X$	$(G, \alpha) \models \forall X\phi \Leftrightarrow$ for all $A \subseteq U$ it holds that $(G, \alpha_X^A) \models \phi$

Herein  $\alpha_x^a$  denotes an assignment with  $\alpha_x^a(\delta) = a$  and  $\alpha_x^a(\zeta) = \alpha(\zeta)$  for all  $\zeta \neq x$ .

Analogously to first order logic, an MSO-sentence is an MSO-formula without free variables. Now, we are ready to present the main result in this field according to fixed-parameter algorithms. To this end, we define the following problem:

**MSO-CHECK**

*Given:* A graph  $G$  and an MSO-sentence  $\varphi$ .

*Question:* Is there an assignment  $\alpha$  such that  $(G, \alpha) \models \varphi$ ?

It is easy to see that one can reduce every graph problem that is based on a graph property, expressible by an MSO-sentence, to MSO-CHECK: Compute an MSO-sentence that expresses the graph property and take the graph and the sentence as input for the MSO-CHECK algorithm. Courcelle developed the following important theorem [Cou90]:

**Theorem 6** (Courcelle’s Theorem). *MSO-CHECK is fixed-parameter-tractable with respect to the combined parameter  $(\text{tw}(G), |\varphi|)$ . Moreover, there is a computable function  $f$  and an algorithm that solves MSO-CHECK in time  $f(\text{tw}(G), |\varphi|) \cdot |G| + O(|G|)$ .*

We end with some simple examples for the application of Theorem 6.

**Examples and extension.** We start with a simple and well-studied graph property: *3-colorability*. A (vertex-)coloring of an undirected graph is a mapping from the set of vertices to a (finite) set of colors, such that no two adjacent vertices have the same color. We say that the graph is 3-colorable if there is a coloring with three colors. The graph property “to be 3-colorable” is expressible with an MSO-sentence:

$$\varphi = \exists C_1 \exists C_2 \exists C_3 \left( \forall x : Vx \rightarrow (C_1x \vee C_2x \vee C_3x) \right) \wedge$$

$$\left( \forall e, \forall a \neq b : (Ee \wedge Iae \wedge Ibe) \rightarrow \neg((C_1a \wedge C_1b) \vee (C_2a \wedge C_2b) \vee (C_3a \wedge C_3b)) \right).$$

Thus, due to Theorem 6 to decide whether a graph is 3-colorable is fixed-parameter tractable with respect to the parameter treewidth.

Arnborg et al. [ALS91] generalized this theorem to the case of “extended monadic second-order logic”. Here, we can additionally make use of set cardinalities. This also includes the optimization problems regarding the set sizes. In this work, we only need the operation “ $\min X : P(X)$ ” that expresses that a specific predicate  $P$  holds for a minimum-size vertex set  $X$ . Independently, Borie et al. [BPT92] obtained similar results, but from a more algorithmic point of view. In their Theorem 3.5, they explicitly say that if a “minimum edge/vertex deletion problem”  $\text{PROB}$  that is based on a graph property which is expressible in MSO-logic, then  $\text{PROB}$  is expressible in extended monadic second-order logic. To become familiar with the minimization extension, we give an easy example here: Let  $G = (V, E)$  be an undirected graph. A *vertex cover* is a subset  $C$  of vertices in  $V$  such that every edge in  $E$  is incident to at least one vertex in  $C$ . The predicate  $\text{VC}(X) := \text{“to be a vertex cover”}$ , with  $X$  being a vertex set, is expressible in MSO-logic:

$$\text{VC}(C) = \forall e : Ee \rightarrow (\exists v : (Cv \wedge Ive)).$$

The optimization variant of vertex cover where one asks for a minimum-size vertex cover is expressible in extended MSO-logic:

$$\min C : \text{VC}(C).$$

We use extended MSO in the next section to investigate the parameterized complexity of  $\text{MINIMUM DEGREE DELETION}$  with respect to the parameter treewidth.

### 5.3.2 MSO expression for Minimum Degree Deletion

In the following, we give a monadic second order sentence to prove the fixed-parameter tractability for  $\text{MINIMUM DEGREE DELETION}$  with respect to the parameter  $t_w := \text{“treewidth of the input graph”}$ . Since  $s_v \geq t_w$  it follows that  $\text{MINIMUM DEGREE DELETION}$  is also fixed-parameter tractable with respect to the parameter  $s_v$ .

We start with an observation that helps us gaining an alternative view on the problem.

**Observation 9.** *Let  $G = (V, E)$  be an undirected graph with treewidth  $t_w$ . Let  $M^*$  be any solution set. Let  $C$  be a tree decomposition of  $G - M^*$  with maximum bag-size  $t_w + 1$  and  $C$  is minimal, that is, it is not possible to obtain another tree decomposition by removing vertices from the bags. It holds that  $w_c$  has degree of at most  $t_w - 1$  in  $G - M^*$ .*

*Proof.* Assume towards a contradiction that  $w_c$  has degree at least  $t_w$  in  $G - M^*$ . Let  $L$  be a leaf bag of  $C$ . There is a vertex  $v_l$  that is only in bag  $L$ , because  $C$  is minimal. Since  $L$  has size  $t_w + 1$  by definition,  $v_l$  can have at most  $t_w$  neighbors; a contradiction to the fact that  $w_c$  is the only vertex of minimum degree in  $G - M^*$ .  $\square$

Inspired by Observation 9 we can formalize  $\text{MINIMUM DEGREE DELETION}$  as follows:

MINIMUM DEGREE DELETION\*

*Given:* An undirected graph  $G = (V, E)$ , a distinguished vertex  $w_c \in V$ , an integer  $k \geq 1$ , and an integer  $i \leq t_w$ .

*Question:* Is there a subset  $K \subseteq V \setminus w_c$  of size  $k$  such that  $w_c$  has exactly  $i$  neighbors not in  $K$  and each other vertex is in  $K$  or has more than  $i$  neighbors not in  $K$ .

It is easy to see that  $(G, k)$  is a yes-instance of MINIMUM DEGREE DELETION if and only if  $(G, k, i)$  is a yes-instance of MINIMUM DEGREE DELETION\* for some  $i \leq t_w$ . Hence, we describe an MSO-sentence  $\varphi$  expressing the graph property given by the question of MINIMUM DEGREE DELETION\*. For the ease of presentation we first describe some of the parts of the sentence: The formula part  $\text{adj}(x, y)$  is satisfiable if and only if  $x$  and  $y$  are adjacent:

$$\text{adj}(x, y) = Vx \wedge Vy \wedge (\exists e(Ee \wedge Ixe \wedge Iye)).$$

The formula part  $\text{iNotKNeighbors}(x, K, i)$  is satisfiable if and only if there are at least  $i$  neighbors of  $x$  which are not contained in the vertex subset  $K$ :

$$\begin{aligned} \text{iNotKNeighbors}(x, K, i) = \\ \exists n_1 \exists n_2 \dots \exists n_i \left( \left( \bigwedge_{1 \leq a \leq i} \text{adj}(x, n_a) \wedge \neg K n_a \right) \wedge \left( \bigwedge_{1 \leq a < b \leq i} n_a \neq n_b \right) \right). \end{aligned}$$

The formula part  $\text{iNotKNeighborsExactly}(x, K, i)$  is satisfiable if and only if there are exactly  $i$  neighbors of  $x$  which are not contained in the vertex subset  $K$ :

$$\begin{aligned} \text{iNotKNeighborsExactly}(x, K, i) = \\ \exists n_1 \exists n_2 \dots \exists n_i \left( \left( \bigwedge_{1 \leq a \leq i} \text{adj}(x, n_a) \wedge \neg K n_a \right) \wedge \left( \bigwedge_{1 \leq a < b \leq i} n_a \neq n_b \right) \right. \\ \left. \wedge \left( \forall n_z \left( \left( \text{adj}(x, n_z) \wedge \left( \bigwedge_{1 \leq b \leq i} n_z \neq n_b \right) \right) \rightarrow K n_z \right) \right) \right). \end{aligned}$$

Putting all together we get:

$$\begin{aligned} \varphi_i = \min K \left( \text{iNotKNeighborsExactly}(w_c, K, i) \right. \\ \left. \wedge \left( \forall x (Vx \wedge x \neq w_c \rightarrow (\text{iNotKNeighbors}(x, K, i+1) \vee Kx)) \right) \right). \end{aligned}$$

It is easy to see that  $\varphi_i$  is an MSO-sentence. By construction there is an assignment  $\alpha$  such that  $(G, \alpha) \models \varphi_i$  if and only if there is a subset  $K \subseteq V$  such that  $w_c$  is the only vertex with minimum degree and  $\deg(w_c) = i$  in  $G - K$ . Due to Theorem 6 it follows that MINIMUM DEGREE DELETION\* is fixed-parameter tractable with respect to the combined parameter  $(t_w, |\varphi_i|)$ . Due to Observation 9 an upper bound for the complexity of each  $\varphi_i$  only depends linear on  $t_w$  (which defines the maximum value for  $i$ ). Thus, we get the following theorem:

**Theorem 7.** MINIMUM DEGREE DELETION is fixed-parameter tractable with respect to the parameter  $t_w := \text{“treewidth of the input graph”}$ .

## 5.4 Size of a feedback vertex set as parameter

In this section, we investigate the parameterized complexity of MINIMUM DEGREE DELETION with respect to the parameter  $s_v$ . Since treewidth is a stronger parameter than  $s_v$ , the fixed-parameter tractability of MINIMUM DEGREE DELETION with respect to the parameter  $s_v$  follows from Theorem 7. This result is mainly a classification. A practicable algorithm is not given through Courcelle's Theorem [Cou09]. However, we investigate an interesting special case of the parameter “size of a feedback vertex set”. More precisely, we show fixed-parameter tractability for MINIMUM DEGREE DELETION with respect to the parameter  $s_v^* :=$  “size of a feedback vertex set that does not contain  $w_c$ ”.

Let  $(G = (V, E), w_c, k)$  be the MINIMUM DEGREE DELETION instance and let  $V_f := \{v_{f_1}, \dots, v_{f_{s_v^*}}\}$  be a concrete feedback vertex set that does not contain  $w_c$ . The following observation bounds the final degree of  $w_c$  in a solution graph from above by the parameter  $s_v$ :

**Observation 10.** *Let  $M^*$  be any solution set. It holds that  $w_c$  has degree of at most  $|V_f|$  in  $G - M^*$ .*

*Proof.* Assume that  $w_c$  has degree at least  $|V_f| + 1$  in the solution graph  $G^* := G - M^*$ . Consider a minimal feedback vertex set  $V_f^*$  of the solution graph  $G^*$ . Due to the minimality of  $V_f^*$ ,  $G^* - V_f^*$  is a forest with at least two vertices. Hence,  $G^* - V_f^*$  contains at least two vertices with degree at most 1. This means  $G^*$  contains at least two vertices with at most  $|V_f| + 1$  neighbors, because each of them has at most  $|V_f^*| \leq |V_f|$  neighbors in the feedback vertex set. Thus,  $w_c$  is not the only with minimum degree; a contradiction.  $\square$

We first introduce a template algorithm **MDD-solv** (see Figure 5.4). It solves MINIMUM DEGREE DELETION by calling a subroutine that solves a slightly modified version of MINIMUM DEGREE DELETION after doing some preprocessing and branching. The preprocessing and branching part **MDD-solv** works as follows:

1. Iterate over all subsets of the feedback vertex set  $V_f$  (of size at most  $k$ ) to fix which of the feedback vertex set vertices belongs to the solution set. Remove these vertices from  $G$  and from  $V_f$  and decrease the parameter accordingly. (lines 2-4)
2. Iterate over all possible values  $i$  for the final degree of  $w_c$  in the solution graph. (line 5)
3. Remove iteratively every vertex ( $\neq w_c$ ) with degree at most  $i$  and decrease the parameter accordingly. (lines 6-9)

Now, we specify **AnnotatedMDD** which called as subroutine in the algorithm **MDD-solv**. In line 10, **AnnotatedMDD** is used to solve the following problem:

---

```

1: procedure MDD-solv( $G, w_c, k, V_f$ )
2:   for each  $V_f^* \subseteq V_f$  with  $|V_f^*| \leq k$  do
3:      $G' := G - V_f^*$ 
4:      $k' := k - |V_f^*|$ 
5:     for  $i := 0$  to  $|V_f|$  do
6:       while there is a vertex  $v \neq w_c$  with degree at most  $i$  do
7:         Remove  $v$  from  $G'$ .
8:          $k' := k' - 1$ 
9:       end while
10:      if AnnotatedMDD( $G', w_c, k', V_f' := V_f \setminus V_f^*, i$ ) then
11:        return 'yes'
12:      end if
13:    end for
14:  end for
15:  return 'no'
16: end procedure

```

Figure 5.4: The algorithm **MDD-solv** solves MINIMUM DEGREE DELETION. The variable  $i$  represents the final degree of  $w_c$  in the solution graph. Due to Observation 10, this final degree is bounded from above by the parameter.

#### ANNOTATED MINIMUM DEGREE DELETION

*Given:* An undirected graph  $G = (V, E)$ , a feedback vertex set  $V_f$ , a distinguished vertex  $w_c \in V$  with  $w_c \notin V_f$ , and two positive integer  $k$  and  $i$ . Every vertex in  $G$  except  $w_c$  has degree at least  $i + 1$ .

*Question:* Is there a subset  $M^* \subseteq V \setminus (V_f \cup \{w_c\})$  of size at most  $k$  such that  $w_c$  has degree  $i$  in  $G - M^*$  and every other vertex from  $G - M^*$  has degree at least  $i + 1$ ?

The preprocessing and branching of **MDD-solv** takes  $O(f(s_v^*) \cdot \text{poly}(|x|))$  time such that fixed-parameter tractability for ANNOTATED MINIMUM DEGREE DELETION with respect to  $s_v^*$  implies fixed-parameter tractability for MINIMUM DEGREE DELETION with respect to  $s_v^*$ . Hence, our goal is to develop a fixed-parameter algorithm for ANNOTATED MINIMUM DEGREE DELETION. To become familiar with the problem, we start with an exponential-time algorithm: The algorithm **Annotated-MDD-XP** in Figure 5.5 solves ANNOTATED MINIMUM DEGREE DELETION by branching over all possible size- $i$  subsets of the neighborhood of  $w_c$  (line 2). The neighbors of  $w_c$  that are not in this subset are removed (lines 3-4). This ensures that  $w_c$  will have final degree  $i$ . The algorithm removes iteratively each vertex with degree at most  $i$  (lines 5-8). This ensures that the remaining graph is a solution graph. Clearly, **Annotated-MDD-XP** solves ANNOTATED MINIMUM DEGREE DELETION, but branching over the neighborhood subsets takes  $O(\binom{n}{i})$  time. A fixed-parameter algorithm needs an improved approach. The following two subsection present two different methods for showing fixed-parameter tractability for ANNOTATED MINIMUM DEGREE DELETION with respect to the parameter  $s_v^*$ .



```

1: procedure ANNOTATED-MDD-XP( $G, w_c, k, V_f$ )
2:   for each  $N_r \subseteq N(w_c)$  with  $|N_r| = i$  do
3:      $M^* := V \setminus N_r$ 
4:     Remove each vertex in  $M^*$  from  $G$ 
5:     while there is a vertex  $v \neq w_c$  with degree at most  $i$  do
6:       Remove  $v$  from  $G$ .
7:        $M^* := M^* \cup \{v\}$ 
8:     end while
9:     if  $M^* \cap V_f = \emptyset$  then
10:      if  $|M^*| \leq k$  then
11:        return 'yes'
12:      end if
13:    end if
14:  end for
15:  return 'no'
16: end procedure

```

Figure 5.5: The XP-algorithm Annotated-MDD-XP solves ANNOTATED MINIMUM DEGREE DELETION in  $O(|x|^{|V_f|})$  time with  $|x|$  being the input size.

### 5.4.1 Integer linear programming

In this section, we present a fixed-parameter algorithm that solves ANNOTATED MINIMUM DEGREE DELETION with respect to the parameter  $s_v^* :=$  “size of a feedback vertex set that does not contain  $w_c$ ” by using the technique of integer linear programming together with a result from Lenstra [Len83]. Consider an instance  $(G = (V, E), V_f, w_c, k, i)$  of ANNOTATED MINIMUM DEGREE DELETION with  $V_f = \{v_1, \dots, v_{s_v^*}\}$ . Without loss of generality we assume that  $w_c$  has final degree at least 2 in the solution graph. Instances with smaller final degree for  $w_c$  can be identified and solved in polynomial time. Furthermore, we use the terms of solution set and nearly-solution set. Note that the following definition of solution set is conform to its general definition in Section 2.3.

**Definition 6.** Let  $(G = (V, E), V_f, w_c, k, i)$  be an instance of ANNOTATED MINIMUM DEGREE DELETION. We say  $M^* \subseteq V$  is a **solution set** if:

1.  $|M^*| \leq k$ ,
2.  $M^* \cap (\{w_c\} \cup V_f) = \emptyset$ ,
3.  $w_c$  has degree  $i$  in  $G - M^*$ ,
4. every vertex  $v \neq w_c$  has degree at least  $i + 1$  in  $G - M^*$ .

We say  $M' \subseteq V$  is a **nearly-solution set** if:

1.  $|M'| \leq k$ ,
2.  $M' \cap (\{w_c\} \cup V_f) = \emptyset$ ,

3.  $w_c$  has degree  $i - 1$  in  $G - M'$ ,
4. every vertex  $v \neq w_c$  has degree at least  $i$  in  $G - M'$ .

We start with an important observation concerning the existence of optimal solution sets. More precisely, we show that if there is an optimal solution set  $M^*$ , then  $M^*$  is detectable in time  $O(f(|V_f|) \cdot \text{poly}(|x|))$  or there is a nearly-solution set that is detectable in time  $O(f(|V_f|) \cdot \text{poly}(|x|))$  for a computable function  $f$  only depending on the parameter value  $|V_f|$  and a polynomial only depending on the input size  $|x|$ .

**Proposition 2.** *Let  $M^*$  be an optimal solution set. It holds that:*

1.  $N(w_c) \setminus (M^* \cup V_f) = \emptyset$ , or
2.  $M^* \subseteq N(w_c)$ , or
3.  $\exists M' \subseteq N(w_c)$  such that  $|M'| \leq |M^*|$  and  $M'$  is a nearly-solution set.

*Proof.* We show if  $N(w_c) \setminus M^*$  contains a vertex that is not a feedback vertex set element and  $M^*$  contains a vertex that is not a neighbor of  $w_c$ , then  $\exists M' \subseteq N(w_c)$  such that  $|M'| \leq |M^*|$  and  $M'$  is a nearly-solution set. We start with a simple partition of  $M^*$  into  $M_y$  and  $M_x$ . Let  $M_y$  be  $M^* \cap N(w_c)$  or in other words the solution set vertices which are neighbors of  $w_c$ . Let  $M_x$  be  $M^* \setminus M_y$  or in other words the solution set vertices which are not neighbors of  $w_c$ . Clearly, it holds that  $M_x \neq \emptyset$ . We build an alternative solution set  $M' = M_y \cup a$  with  $a$  being a neighbor of  $w_c$  but not already in  $M_y$  and  $a \notin V_f$ . The vertex  $a$  exists, because  $w_c$  has degree at least 2 in  $G - M^*$  and  $N(w_c) \setminus (M^* \cup V_f) \neq \emptyset$ . First, we show that  $M'$  is at most as big as  $M^*$ . Since  $M_x \neq \emptyset$ ,  $|M_x| \geq 1$ . Thus,  $|M'| = |M^*| - |M_x| + 1 \leq |M^*|$ . Second, we show that  $M'$  is a nearly-solution set by showing each point of the definition:

1. Since  $|M^*| \leq k$  and  $|M'| \leq |M^*|$ , it holds that  $|M'| \leq k$ .
2. The set  $M'$  contains only one vertex that is not also in  $M^*$ , namely  $a$ . Since  $a \neq w_c$  and  $a \notin V_f$ , it holds that  $M' \cap (\{w_c\} \cup V_f) = \emptyset$ .
3. By definition  $w_c$  has degree  $i$  in  $G - M^*$ . Since  $M'$  contains only one neighbor of  $w_c$  that is not also in  $M^*$ , it holds that  $w_c$  has degree  $i - 1$  in  $G - M'$ .
4. Each vertex from  $V \setminus M^*$  except  $w_c$  has degree at least  $i + 1$  in  $G - M^*$ . The additional vertex  $a$  can decrease the degree of each vertex from  $V \setminus M^*$  at most by one. Hence, each vertex from  $V \setminus M^*$  except  $w_c$  has degree at least  $i$  in  $G - M'$ . It remains to show that the degree of each vertex from  $M_x$  is at least  $i$  in  $G - M'$ , too. Clearly, each vertex from  $M_x$  has degree  $i + 1$  in the input graph  $G$ . Assume that  $\exists x \in M_x$  with  $\deg(x) < i$  in  $G - M'$ . Thus,  $x$  must have two neighbors in  $M'$ . Furthermore,  $x$  must be adjacent to two distinct vertices  $b, c \in N(w_c)$ , because  $M' \subseteq N(w_c)$ . Since  $\{x, b, c, w_c\} \cap V_f = \emptyset$ , there is a cycle  $G[\{x, b, c, w_c\}]$  in the forest  $G - V_f$ ; a contradiction.

□

**Coefficients:**

- $\tilde{v}_j$  denotes the number of neighbors of  $v_j \in V_f$  that are not in  $N(w_c)$
- $\tilde{g}_j$  denotes the number of group- $g_j$  vertices in  $N(w_c)$ .
- $\hat{g}_j$  denotes the number of group- $g_j$  vertices in  $N(w_c)$  that have a neighbor  $x \neq w_c$  with  $\deg(x) = i + 1$

The variable  $x_j$  denotes the number of group- $g_j$  vertices in  $N(w_c)$  that are not part of the solution set.

**ILP-1****minimize:**

$$\sum_{1 \leq j \leq 2^{s_v^*}} x_j$$

**subject to**

$$\begin{aligned} &\text{for all } 1 \leq j \leq 2^{s_v^*} : \hat{g}_j \leq x_j \leq \tilde{g}_j \\ &\text{for all } 1 \leq q \leq s_v^* : \tilde{v}_q + \sum_{j \text{ with } v_q \in G_j} x_j \geq i + 1 \end{aligned}$$

**ILP-2****minimize:**

$$\sum_{1 \leq j \leq 2^{s_v^*}} x_j$$

**subject to**

$$\begin{aligned} &\text{for all } 1 \leq j \leq 2^{s_v^*} : x_j \leq \tilde{g}_j \\ &\text{for all } 1 \leq q \leq s_v^* : \tilde{v}_q + \sum_{j \text{ with } v_q \in G_j} x_j \geq i \end{aligned}$$

Figure 5.6: Integer linear programs “ILP-1” and “ILP-2”. There is an optimal solution set  $M^1 \subseteq N(w_c)$  if and only if the objective of ILP-1 is at most  $i$  and a nearly-solution set  $M^2 \subseteq N(w_c)$  if and only if the objective of ILP-2 is at most  $i - 1$ .

Due to Proposition 2, one can determine whether  $(G, V_f, w_c, k, i)$  is a yes-instance of ANNOTATED MINIMUM DEGREE DELETION by checking each of the three cases. The first case is quite simple. Since all neighbors of  $w_c$  which are not in the optimal solution set are feedback vertex set elements and the solution set does not contain any feedback vertex set element, the optimal solution can be found as follows: Firstly, remove  $N(w_c) \setminus V_f$  from  $G$ . The remaining solution set vertices are determined by iteratively removing every vertex with degree at most  $i$ . Finally, if one did not remove any vertex from  $V_f$  and the total number of removed vertices is at most  $k$ , then  $(G, V_f, w_c, k, i)$  is a yes-instance.

The remaining two cases can be handles by two quite similar integer linear programs. We briefly discuss the two cases and develop the integer linear programs in Figure 5.6. For the second case it remains to search for a set  $M^1 \subseteq N(w_c)$  of size at most  $k$  such that  $w_c$  is the only degree- $i$  vertex in  $G - M^1$  and every other vertex has degree at least  $i + 1$ . For the third case it remains to search for a set  $M^2 \subseteq N(w_c)$  of size at most  $k$  such that  $w_c$  is the only degree- $i - 1$  vertex in  $G - M^2$  and every other vertex has degree at least  $i$ . If  $M^1$  or  $M^2$  exists, then  $(G, V_f, w_c, k, i)$  is a yes-instance. To determine whether  $M^1$  or  $M^2$  exists, we consider the input graph  $G = (V, E)$ . Every vertex in  $G$  (except  $w_c$ ) has degree at least  $i + 1$ . Clearly, no vertex in  $M^1$  can have a neighbor with degree  $i + 1$  in  $G$  (except  $w_c$ ). In contrast,  $M^2$  may contain also vertices with degree- $i + 1$  neighbors. The main question is, which neighbors of  $w_c$  are in  $M^1$  respectively in  $M^2$ .

To this end, we consider the dual question: Which neighbors of  $w_c$  are not in  $M^1$  respectively in  $M^2$ . Formally, we denote  $N^1 := N(w_c) \setminus M^1$  and  $N^2 := N(w_c) \setminus M^2$  as the remaining neighbors of  $w_c$ . We have to ensure that each vertex from  $V_f$  has final degree at least  $i + 1$  in  $G - M^1$  respectively final degree at least  $i$  in  $G - M^2$ . Hence, the  $V_f$ -neighborhood of a vertex in  $N(w_c)$  is important to decide whether the vertex is in  $N^1$  respectively  $N^2$ . Of course, each of the vertices in  $V_f$  can have neighbors that are not in  $N(w_c)$ . To express this, we need the following definition:

**Definition 7.** We denote the number of neighbors of  $v_j \in V_f$  that are not in  $N(w_c)$  as  $\tilde{v}_j$ .

The remaining neighbors of each feedback vertex set element must be in  $N^1$  respectively  $N^2$ . Moreover, we can group the vertices in  $N(w_c)$  by the subsets of vertices in  $V_f = \{v_1, \dots, v_{s_v^*}\}$  that are connected to them. Note that in the case of searching for  $N^1$ , we already know that each vertex that has a neighbor  $x \neq w_c$  with  $\deg(x) = i + 1$  must be in  $N^1$ .

**Definition 8.** Let  $\{G_1, \dots, G_{2s_v^*}\}$  be the subsets of  $V_f$ . We say a vertex  $v$  is a group- $g_j$  vertex if  $N(v) \cap V_f = G_j$ . Furthermore,  $\tilde{g}_j$  denotes the number of group- $g_j$  vertices in  $N(w_c)$  and  $\hat{g}_j$  denotes the number of group- $g_j$  vertices in  $N(w_c)$  that have a neighbor  $x \neq w_c$  with  $\deg(x) = i + 1$ .

Besides the existence of a degree- $i + 1$  neighbor, it is not important which vertex of each group is contained in  $N^1$  respectively  $N^2$ . Only the total number of vertices in  $N^1$  respectively  $N^2$  must be as small as possible under the condition that the final degree of each feedback vertex set element is at least  $i + 1$  respectively at least  $i$ .

Summarizing, in the second case, we have to find a minimum-size set  $N^1 \subseteq N(w_c)$  by determining  $x_j$ , that is, the number of vertices of each group  $g_j$  that are in  $N^1$  for  $j \in \{1, \dots, 2^{s_v^*}\}$ . The first condition is that from each group  $g_j$  at least  $\hat{g}_j$  and at most  $\tilde{g}_j$  vertices must be in  $N^1$ . The second condition is that for each feedback vertex set element  $v_l$  the sum of  $x_j$  with  $v_l \in G_j$  plus  $\tilde{v}_j$  exceeds  $i + 1$ . Clearly, if the minimal sum of  $x_j$  with  $j \in \{1, \dots, 2^{s_v^*}\}$  is at most  $i$ , the corresponding sets  $N^1$  and  $M^1$  exists as well.

In the third case, we have to find a minimum-size set  $N^2 \subseteq N(w_c)$  by determining  $x_j$ , that is, the number of vertices of each group  $g_j$  that are in  $N^1$  for  $j \in \{1, \dots, 2^{s_v^*}\}$ . The first condition is that from each group  $g_j$  at most  $\tilde{g}_j$  vertices must be in  $N^1$ . The second condition is that for each feedback vertex set element  $v_l$  the sum of  $x_j$  with  $v_l \in G_j$  plus  $\tilde{v}_j$  exceeds  $i$ . Clearly, if the minimal sum of  $x_j$  with  $j \in \{1, \dots, 2^{s_v^*}\}$  is at most  $i - 1$ , the corresponding sets  $N^2$  and  $M^2$  exists as well. Both tasks are formulated as integer linear programs (see Figure 5.6).

Due to results of Lenstra [Len83] and Kannan [Kan87] we use the following theorem to show fixed-parameter tractability for the computation of the integer linear program results:

**Theorem 8** (Lenstra’s theorem). *INTEGER LINEAR PROGRAMMING FEASIBILITY can be solved with  $O(p^{9p/2} \cdot L)$  arithmetic operations in integers of  $O(p^{2p} \cdot L)$  bits in size, where  $p$  is the number of ILP variables and  $L$  is the number of bits in the input.*

The running time of the integer linear program is in  $O(|V_f|^{\frac{9}{2} \cdot |V_f|} \cdot \text{poly}(|x|))$  time. Putting all together, we arrive at the following:

**Theorem 9.** *The problem ANNOTATED MINIMUM DEGREE DELETION can be solved in  $O(s_v^{*\frac{9}{2} \cdot s_v^*} \cdot \text{poly}(|x|))$  time with  $s_v^*$  being the size of a feedback vertex set that does not contain  $w_c$  and  $|x|$  being the size of the input.*

### 5.4.2 Dynamic programming

In this section, we present a fixed-parameter algorithm that solves ANNOTATED MINIMUM DEGREE DELETION with respect to the parameter  $s_v^* :=$  “size of a feedback vertex set that does not contain  $w_c$ ” by using the technique of dynamic programming. Let  $(G = (V, E), V_f, w_c, k, i)$  be a ANNOTATED MINIMUM DEGREE DELETION instance with  $V_f := \{v_1, \dots, v_{s_v^*}\}$ . We start with some general observation and define basic concepts. By definition, every solution set contains only vertices in  $V_S := V \setminus (V_f \cup \{w_c\})$ . The following observation considers the neighborhood of the vertices in  $V_S$ .

**Definition 9.** *Let  $G = (V, E)$  be an undirected graph and  $x \in V_S$  a vertex. We denote  $\text{Component}_{G[V_S]}(x)$  as the connected component from  $G[V_S]$  including the vertex  $x$ .*

**Observation 11.** *Let  $n_a, n_b$  be two distinct neighbors of  $w_c$ . It holds that:*

$$\text{Component}_{G[V_S]}(n_a) \cap \text{Component}_{G[V_S]}(n_b) = \emptyset$$

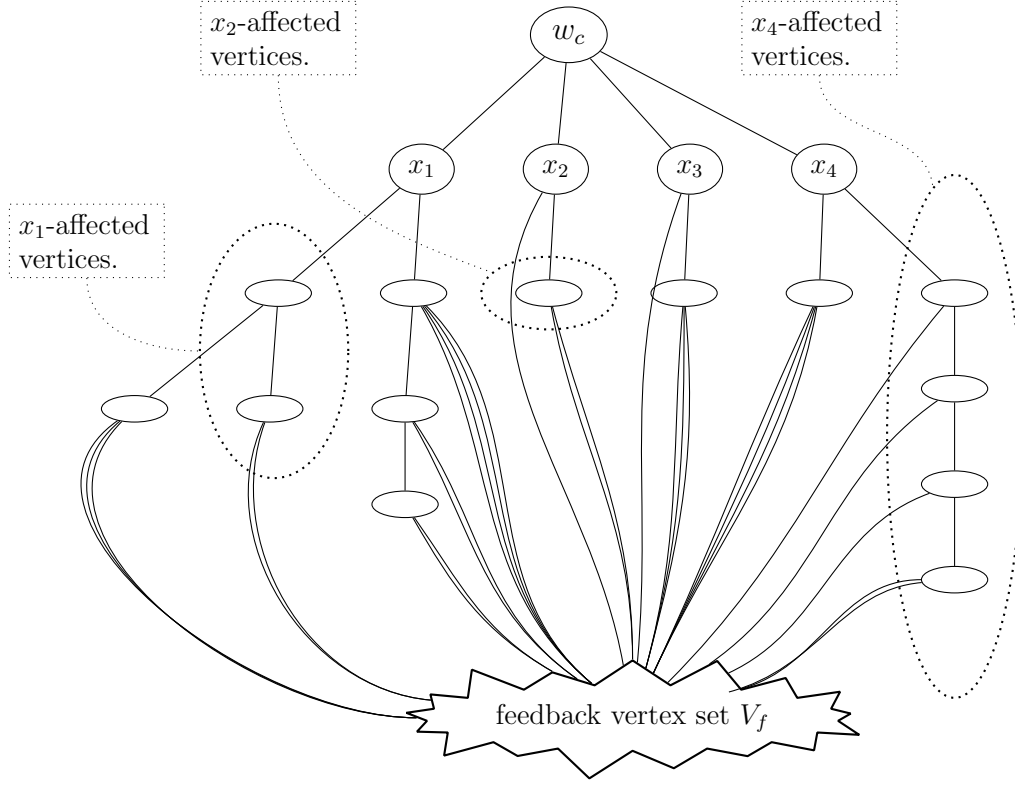


Figure 5.7: Affected and unaffected vertices. Let  $i = 2$  denote the final degree of  $w_c$  given by the input. For example, if one deletes  $x_1$ , then one has also to delete the marked  $x_1$ -affected vertices.

*Proof.* Assume that there is an edge between one vertex  $v_a$  of  $\text{Component}_{G[V_S]}(n_a)$  and another vertex  $v_b$  of  $\text{Component}_{G[V_S]}(n_b)$  in  $G[V_S]$ . In  $G - V_f$  both components are connected through  $w_c$ . Thus,  $G[\{v_a, w_c, v_b\}]$  is a  $C_3$  and  $G - V_f$  is cyclic; a contradiction.  $\square$

Now, we take a look at an optimal solution set  $M^*$ : Clearly,  $M^*$  must contain  $\deg(w_c) - i$  neighbors of  $w_c$ . In addition, putting a vertex  $x \in N(w_c)$  into a solution set can decrease the degree of other vertices from  $\text{Component}_{G[V_S]}(x)$ . Hence, removing  $x$  can enforce to remove further vertices from  $\text{Component}_{G[V_S]}(x)$  recursively. To measure this, we need the following definition which is illustrated in Figure 5.7:

**Definition 10.** We denote  $A[x]$  as the  $x$ -affected vertices, that is, the set of vertices that have to be removed when removing  $x$ . Furthermore, we denote  $\text{cost}(x) := |A[x]|$  as the number of vertices that have to be removed when removing  $x$ .

To determine a solution set, one is interested in a set consisting of  $\deg(w_c) - i$  neighbors of  $w_c$  such that the sum of the corresponding costs is minimal. The critical point is that it is also necessary to “measure” the effect that putting a vertex  $x$  into the solution set has to the vertices from  $V_f$ . By definition, we can not remove any vertex from  $V_f$ . Thus, we must ensure that the final degree of every vertex from  $V_f$  is at least  $i + 1$ . In the following we consider a partition of  $V_S$  into a set  $V_u$  containing vertices that must not belong to the solution and  $V_r$  containing vertices

**Require:**

- $G$  is an undirected graph such that each vertex (except  $w_c$ ) must have degree at least  $i + 1$
- $V_f$  is a feedback vertex set of  $G'$  that does not contain  $w_c$

```

1: procedure ANNOTATEDMDD( $G, w_c, k, V_f, i$ )
2:   Compute to-remain-tuple  $S^0 := (s_1^0, \dots, s_{|V_f|}^0)$  ▷ Initialization
3:   for  $z = 1$  to  $\deg(w_c)$  do
4:     for each  $S' \in \mathcal{S}$  do
5:        $T(0, z, D) = +\infty$ 
6:     end for
7:   end for
8:   for each  $S' = (s'_1, s'_2, \dots, s'_{|V_f|}) \in \mathcal{S}$  do
9:     if  $s'_j < s_j^0$  for any  $j \in \{1, \dots, |V_f|\}$  then
10:       $T(0, 0, S') = +\infty$ 
11:    else
12:       $T(0, 0, S') = 0$ 
13:    end if
14:  end for
15:  for  $x = 1, \dots, |N(w_c)|$  do ▷ Table update
16:    for  $z = 1, \dots, x$  do
17:      for each  $S' \in \mathcal{S}$  do
18:         $\text{minCostRemove} := T(x - 1, z - 1, S') + \text{cost}_i(n_x)$ 
19:         $\text{minCostNotRemove} := T(x - 1, z, \text{Remain}(S', n_x))$ 
20:         $T(x, z, S') := \min(\text{minCostRemove}, \text{minCostNotRemove})$ 
21:      end for
22:    end for
23:  end for
24:  if  $T(\deg(w_c), \deg(w_c) - i, (0, \dots, 0)) \leq k$  then ▷ Result
25:    return 'yes'
26:  else
27:    return 'no'
28:  end if
29: end procedure

```

**Ensure:** Returns yes if and only if there is a solution set  $M^*$  such that

- $|M^*| \leq k$
- $V_f \cap M^* = \emptyset$
- $w_c$  has degree exactly  $i$  in  $G - M^*$

Figure 5.8: AnnotatedMDD.

that might be part of the solution. For each vertex  $v \in V_f$  one can compute “how many neighbors from  $V_r$  are not allowed to be deleted”. To this end, we define a tuple expressing the effect of  $V_u$  to the hole feedback vertex set:

**Definition 11.** We define the **to-remain-tuple** for  $V_r \subseteq V_S$  as  $S = (s_1, \dots, s_{|V_f|})$  where  $s_j$  denotes the minimum number of  $V_r$ -neighbors of  $v_j$  that are not allowed to be deleted.

Now, we describe the dynamic programming algorithm **AnnotatedMDD** (see Figure 5.8). Basically the algorithm iterates over neighbor subsets of  $N(w_c)$  and computes for every such subset the minimum cost under the condition that deleting a number of vertices from  $N(w_c)$  results in a specific to-remain-tuple. More specifically, the dynamic programming table is defined as  $T(x, z, S')$  with:

- $x \in \{1, \dots, |N(w_c)|\}$ ,
- $z \leq x$ , and
- $S' \subseteq \mathcal{S} := \{(s'_1, \dots, s'_{|V_f|}) \mid 0 \leq s'_i \leq i + 1\}$

The entry  $T(x, z, S')$  then contains the minimum cost of deleting a size- $z$  subset  $N'$  with  $N' \subseteq \{n_i \in N(w_c) \mid i \leq x\}$  that “realizes” the to-remain-tuple  $S'$  for  $N'_r := N(w_c) \setminus N'$ . By definition of the table,  $T(\deg(w_c), \deg(w_c) - i, (0, \dots, 0)) \leq k$  if and only if  $(G, V_f, w_c, k, i)$  is a yes-instance of **ANNOTATED MINIMUM DEGREE DELETION**.

In the following, we describe the details of the algorithm **AnnotatedMDD** and show its correctness. To state the **Remain()** function in line 19 (see Figure 5.8), we need a function that computes the modification of a to-remain-tuple when the algorithm fixes a specific neighbor of  $w_c$  to be not contained in the solution set.

**Definition 12.** Let  $S' := \{s'_1, \dots, s'_{|V_f|}\}$  be a to-remain-tuple for a subset  $V'_r \subseteq V_S$  and  $x \subseteq (N(w_c) \cap V'_r)$ . We define  $\text{Remain}(S', x) := (r_1, \dots, r_{|V_f|})$  where  $r_j$  denotes the minimum number of  $V'_r$ -neighbors of  $v_j$  that are not allowed to be deleted with  $V''_r := V'_r \setminus A[x]$ . More specifically:

$$r_j := \min\{\max\{0, s'_j + e(v_j, \{x\})\}, i + 1\}$$

with  $e(v_j, x)$  denoting the number of neighbors of a feedback vertex set element  $v_j$  in  $A[x]$ .

Furthermore, a kind of “upper bound” for the to-remain-tuple is given: Let  $U$  denote the *unaffected vertices*, that is, those vertices that are not affected by removing any neighbor of  $w_c$ . The to-remain-tuple  $S^0 := (s^0_1, \dots, s^0_{|V_f|})$  for  $V_S \setminus U$  is given by:

$$s^0_j := \max\{0, i + 1 - |N(v_j) \cap U|\}$$

**Lemma 7.** *AnnotatedMDD is correct.*



*Proof.* Consider the algorithm in Figure 5.8. It starts with the computation of  $S^0$  (line 2). In the initialization, the cost of “removing at least one vertex from a set of zero vertices” must be set to infinity (lines 3-7). Furthermore, without fixing any neighbor to be “not contained in the solution set” it is not possible to realize a to-remain-tuple that is better than  $S^0$  which means that there is an entry with a smaller value. Clearly, initializing these table values with infinity is correct (lines 9-10). By definition, it is easy to realize a to-remain-tuple  $S'$  which is worse than  $S^0$ , that is, every entry  $S'$  is at least as big as in  $S^0$ . Therefore one has no costs (lines 11-12). Now, consider the update step. It is easy to verify that the three loops (lines 15-17) ensure that every previous value that is used for computation (lines 18-19) has been computed before. For showing correctness it remains to prove the correctness of this computation. Consider a table entry  $T(x, z, S')$ . It contains the minimum costs for removing  $z$  vertices from  $N' \subseteq \{n_i \in N(w_c) \mid i \leq x\}$ . Regarding the neighbor  $n_x$ , either it is part of the solution or not. If  $n_x$  is removed, then the minimum costs are exactly the minimum costs for removing  $z - 1$  vertices from  $N' \subseteq \{n_i \in N(w_c) \mid i \leq (x - 1)\}$  plus  $\text{cost}(n_x)$  (line 18). Otherwise, if  $n_x$  is not removed, then the costs are exactly the minimum costs for removing  $z$  vertices from  $N' \subseteq \{n_i \in N(w_c) \mid i \leq (x - 1)\}$ , but realizing the to-remain-tuple under the condition that  $n_x$  is fixed to be not part of the solution (line 19).  $\square$

Now, we analyze the running time and table size:

**Lemma 8.** *The table-size of  $T$  is bounded by  $O((|V_f|+2)^{|V_f|} \cdot \deg(w_c)^2)$ . The running time of *AnnotatedMDD* is bounded by a function in  $O((|V_f|+2)^{|V_f|} \cdot \deg(w_c)^2 \cdot n^2)$ .*

*Proof.* The first two dimensions are easily bounded by  $\deg(w_c)$ . The to-remain-tuple is only defined such that each of the  $|V_f|$  entries has an integer value between 0 and  $|V_f| + 1$  (see Definition 11). Hence,  $|\mathcal{S}| = (|V_f| + 2)^{|V_f|}$ . Clearly, the remaining steps can be accomplished in  $O(n^2)$  time.  $\square$

Together with the preprocessing steps of the algorithm *MDD-solv* in Figure 5.4 we arrive at the following:

**Theorem 10.** *MINIMUM DEGREE DELETION can be solved in  $O((s_v^* + 2)^{s_v^*} \cdot 2^{s_v^*} \cdot n^4 \cdot \deg(w_c)^2)$  with  $s_v^*$  being the size of a feedback vertex set that does not contain  $w_c$ .*

The additional factor of  $2^{|V_f|} \cdot n^2$  is due to the outer loops of *MDD-solv* (lines 2-9).

## 5.5 No polynomial kernel with respect to $s_v$

In the previous sections, we presented several fixed-parameter tractability results for MINIMUM DEGREE DELETION with respect to the parameters  $t_w :=$  “treewidth of the input graph”,  $s_v :=$  “size of a feedback vertex set”, and  $s_v^* :=$  “size of a feedback vertex set that does not contain  $w_c$ ”. These fixed-parameter tractability results imply problem kernels. However, such kernels can have exponential (or even greater) sizes. For theoretical and of course also for practical reasons one is interested in problem kernels of small sizes. Although we showed a vertex-linear problem kernel with respect to the parameter  $s_e :=$  “size of a feedback edge set”, we did not find at

least a polynomial kernel for  $t_w$ ,  $s_v$ , and  $s_v^*$ . In this section, we show that (unless the polynomial-time hierarchy [Sto76] collapses at third level) there is no polynomial kernel for MINIMUM DEGREE DELETION even with respect to the parameter  $s_c^* := \text{“size of a vertex cover that does not contain } w_c\text{”}$ . Since every vertex cover is also a feedback vertex set,  $s_c^*$  is a weaker parameter than  $s_v^*$  which is clearly a weaker parameter than  $s_v$ . Hence, the non-existence of the polynomial kernel can be carried over to  $t_w$ ,  $s_v$  and  $s_v^*$ .

Bodlaender et al. [BDFH09] and Fortnow et al. [FS08] developed a framework for showing the existence or non-existence of a polynomial kernel. We need the following definition:

**Definition 13.** ([BDFH09, FS08]) A **composition algorithm** for a parameterized problem  $L \subseteq \Sigma \times \mathbb{N}$  is an algorithm that receives as input a sequence  $((x_1, k), \dots, (x_t, k))$ , with  $(x_i, k) \in \Sigma \times \mathbb{N}^+$  for each  $1 \leq i \leq t$ , uses time polynomial in  $\sum_{i=1}^t |x_i| + k$ , and outputs  $(y, k') \in \Sigma \times \mathbb{N}^+$  with:

1.  $(y, k') \in L \Leftrightarrow (x_i, k) \in L$  for some  $1 \leq i \leq t$ , and
2.  $k'$  is polynomial in  $k$ .

A parameterized problem is called **compositional**, if there exists a composition algorithm.

**Theorem 11.** ([BDFH09, FS08]) If any compositional problem whose unparameterized version is NP-complete has a polynomial kernel, then  $\text{coNP} \subseteq \text{NP} / \text{poly}$ .

#### Note 1

Theorem 11 implies  $\text{PH} = \Sigma_p^3$  [Yap83], but Cai et al. [CCHO05] improved this implication to the collapse  $\text{PH} = S_2^{\text{NP}}$ , which is even a stronger result. However, the weaker collapse to  $\Sigma_p^3$  may be more familiar.

Bodlaender et al. [BDFH09] introduced a refined concept of parameterized reduction that allows to transfer non-kernelizable results to new problems:

**Definition 14.** ([BTY08]) Let  $P$  and  $Q$  be parameterized problems. We say that  $P$  is **polynomial time and parameter reducible** to  $Q$ , written  $P \leq_{\text{ptp}} Q$ , if there exists a polynomial time computable function  $f : \Sigma \times \mathbb{N} \rightarrow \Sigma \times \mathbb{N}$  and a polynomial  $p$ , such that for all  $(x, k) \in \Sigma \times \mathbb{N}$ :

1.  $(x, k) \in P \Leftrightarrow (x', k') = f(x, k) \in Q$ , and
2.  $k' \leq p(k)$ .

The function  $f$  is called **polynomial time and parameter transformation**.

**Proposition 3.** ([BTY08]) Let  $P$  and  $Q$  be parameterized problems, and suppose that  $P^c$  and  $Q^c$  are the derived classical problems. Suppose that  $Q^c$  is NP-complete, and  $P^c \in \text{NP}$ . Suppose that  $f$  is a polynomial time and parameter transformation from  $P$  to  $Q$ . Then, if  $Q$  has a polynomial kernel, then  $P$  has a polynomial kernel.

Each vertex  $x \in \{u_1, \dots, u_d\} \cup \{s_1, \dots, s_n\}$  is connected to exactly  $k$  vertices in  $C$ .

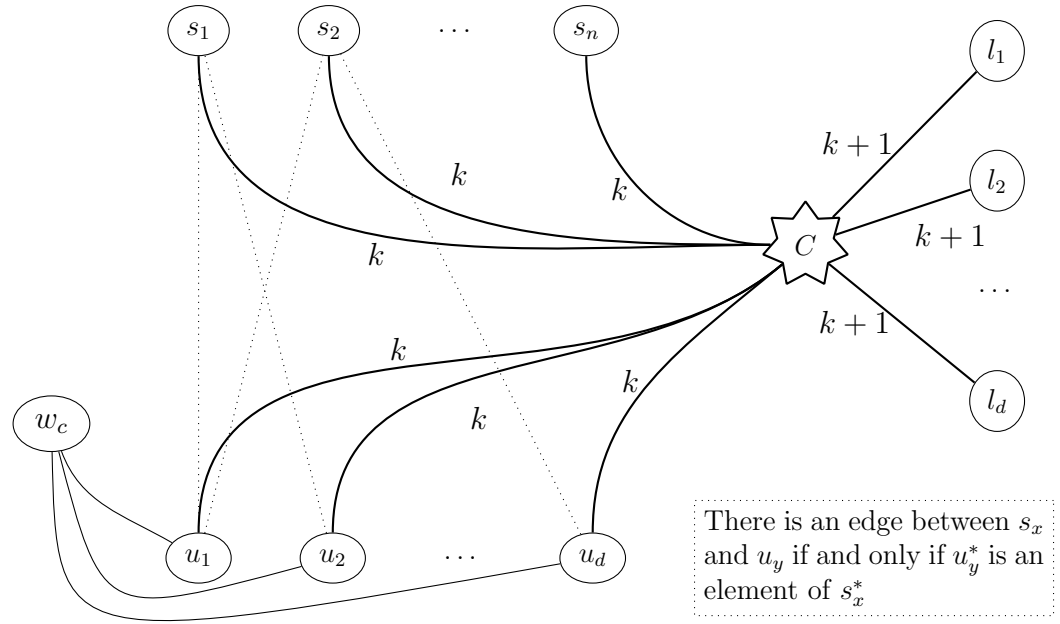


Figure 5.9: MINIMUM DEGREE DELETION instance  $(G, w_c, k)$  obtained from a polynomial time and parameter transformation from a SMALL UNIVERSE HITTING SET instance  $(U^*, S^*, k)$  with universe-size  $d$ . The star  $C$  represents a clique with  $k + 1$  vertices. A fat line that is labeled with a weight  $j$  represents  $j$  edges between a clique and a vertex. Dotted lines between two vertices  $u_i$  and  $s_j$  represent a single edge which states that  $u_i^*$  is an element of  $s_j^*$ .

Now, we show a polynomial time and parameter transformation from SMALL UNIVERSE HITTING SET with respect to the combined parameter  $d :=$ “size of the universe” and  $k :=$ “solution size” to MINIMUM DEGREE DELETION with respect to the combined parameter  $s_c^* :=$ “size of a vertex cover that does not contain  $w_c$ ” and  $k' :=$ “solution size”.

#### SMALL UNIVERSE HITTING SET

*Given:* A family  $S$  over a universe  $U$  with  $|U| \leq d$  and a positive integer  $k$ .

*Question:* Is there a subset  $U' \subseteq U$  of size at most  $k$  such that every set in  $S$  has a non-empty intersection with  $U'$ ?

Dom et al. [DLS09] showed that SMALL UNIVERSE HITTING SET with respect to the parameter  $(d, k)$  does not have a polynomial kernel unless the polynomial-time hierarchy collapses. In fact, they showed that a colored version of SMALL UNIVERSE HITTING SET is compositional and there is a polynomial time and parameter transformation from the colored to the uncolored version.

**Polynomial time and parameter transformation.** This transformation is illustrated in Figure 5.9. Let  $(U^*, S^*, k)$  be a SMALL UNIVERSE HITTING SET instance with a universe  $U^* = \{u_1^*, \dots, u_d^*\}$ , the subset family  $S^* = \{s_1^*, \dots, s_n^*\}$ , and the size of the solution set  $k$ . We construct an undirected graph  $G$  with a distinguished vertex  $w_c$ , such that  $(G, w_c, k' := d - k)$  is a yes-instance of MINIMUM DEGREE DELETION if and only if  $(U^*, S^*, d, k)$  is a yes-instance of SMALL UNIVERSE HITTING SET. First, we create for each element of the universe  $u_i^*$  one vertex  $u_i$  and for each subset  $s_j^*$  in  $S^*$  one vertex  $s_j$ . There is an edge between  $u_i$  and  $s_j$  if and only if  $u_i^*$  is an element in  $s_j^*$ . Additionally, we create the vertex  $w_c$  which should become the vertex with minimum degree. Our goal is to ensure that an optimal solution deletes  $d - k$  vertices from  $U := \{u_1, \dots, u_d\}$  and no other vertex. Therefore we set the degree of  $w_c$  to  $d$  and for each other vertex to at least  $k$ . To this end, we create one clique  $C$  of size  $k + 1$  and a set of  $d$  vertices  $L = \{l_1, \dots, l_d\}$ , where each of them is connected to each vertex in  $C$ . Each vertex that is in  $U$  or in  $S := \{s_1, \dots, s_n\}$  is connected to exactly  $k$  vertices in  $C$ . Thus,  $w_c$  has degree  $d$  and each other vertex has degree at least  $k + 1$ .

We are now ready to prove the correctness of the polynomial time and parameter transformation. The first direction of the equivalence is quite simple.

**Lemma 9.** *If  $(U^*, S^*, k)$  is a yes-instance of SMALL UNIVERSE HITTING SET then  $(G, w_c, k')$  is a yes-instance of MINIMUM DEGREE DELETION.*

*Proof.* Let  $U' \subseteq U^*$  with  $|U'| = k$  be a solution set of  $(U^*, S^*, d, k)$ , that is, each subset  $\in S^*$  has a non-empty intersection with  $U'$ . The set  $M := \{u_j \mid u_j^* \in U^* \setminus U'\}$  is a solution set of  $(G, w_c, d - k)$ : The vertex  $w_c$  has degree  $k$  in  $G - M$ , since  $U^*$  has size  $d$  and  $U'$  has size  $k$ . By the construction of  $G$  each other vertex has degree at least  $k + 1$ . Thus,  $(G, w_c, d - k)$  is a yes-instance of MINIMUM DEGREE DELETION.  $\square$

The proof of the other direction of the equivalence needs some observations. In the following, let  $M_d$  be any solution set.

**Observation 12.** *The vertex  $w_c$  has degree  $k$  in  $G - M_d$ .*

*Proof.* Assume that  $w_c$  has degree more than  $k$  in  $G - M_d$ . Since  $w_c$  is the only vertex with minimum degree in  $G - M_d$ ,  $M_d$  must contain every vertex in  $L$ , because each vertex in  $L$  has degree  $k + 1$ . That means  $M_d$  is of size  $d > d - k$ ; a conflict. Assume that  $w_c$  has degree less than  $k$  in  $G - M_d$ . In this case,  $M_d$  must contain at least  $d - (k - 1)$  neighbors of  $w_c$ ; a conflict.  $\square$

**Observation 13.** *The graph  $G$  has a vertex cover of size  $k + 1 + d$  which does not contain  $w_c$ .*

*Proof.* The set  $C \cup U$  is a vertex cover:  $G - (C \cup U)$  does not contain any edge;  $C$  is of size  $k + 1$  and  $U$  of size  $d$ .  $\square$

**Lemma 10.** *If  $(G, w_c, k')$  is a yes-instance of MINIMUM DEGREE DELETION then  $(U^*, S^*, k)$  is a yes-instance of SMALL UNIVERSE HITTING SET.*

*Proof.* It remains to show that there is a hitting set  $U' \subseteq U^*$ . One can build  $U'$  as follows: For each vertex  $u_i \in U$  which is not in  $M_d$  add the element  $u_i^*$  to  $U'$ . Due to Observation 12 the size of  $U'$  is  $k$ . It remains show that  $U'$  is a hitting set. Assume that there is a subset  $s_j^*$  with  $j \in \{1, \dots, n\}$  that has no intersection with any element in  $U'$ . Thus, for each element  $u_i^* \in s_j^*$  the corresponding vertex  $u_i$  is in  $M_d$ . Due to the construction of  $G$  the vertex  $s_j$  has degree  $k$  in  $G - M_d$ . It follows that  $M_d$  is not a solution set; a conflict.  $\square$

**Lemma 11.** *There is a polynomial time and parameter transformation from SMALL UNIVERSE HITTING SET with respect to the combined parameter  $(d, k)$  to MINIMUM DEGREE DELETION with respect to the combined parameter  $(s_c^*, k')$ .*

*Proof.* Due to Lemma 9 and 10 the equivalence of both instances (point 1 of Definition 14) is given. Due to Observation 13 and the construction of graph  $G$ , the new parameter  $(s_c^*, k')$  is bounded by a polynomial only depending on the old parameter  $(d, k)$  (point 2 of Definition 14):

$$(s_c^*, k') = (d + 1 + k, d - k).$$

$\square$

Putting all together, we arrive at the following theorem:

**Theorem 12.** *MINIMUM DEGREE DELETION has no polynomial kernel with respect to the combined parameter  $(s_c^*, k)$ , with  $s_c^*$  being the size of a vertex cover that does not contain the distinguished vertex and  $k$  being the number of to deleted vertices, unless  $\text{coNP} \subseteq \text{NP} / \text{poly}$ .*

Of course, this implies that there is no hope for polynomial kernels for the parameters  $t_w$ ,  $s_v$ ,  $s_v^*$  and  $s_c :=$ “size of a vertex cover” as single parameters, or combined with  $k$ .



## 6 Bounded Degree Deletion

In this chapter, we analyze the parameterized complexity of BOUNDED DEGREE DELETION. The problem is motivated as graph problem where one searches for a vertex subset of size at most  $k$  whose removal from the graph is a graph in which each vertex has degree at most  $d$ . The following section summarizes known results of the parameterized complexity of BOUNDED DEGREE DELETION which are obtained from [Mos10].

### 6.1 Known results

BOUNDED DEGREE DELETION is fixed-parameter tractable with respect to the parameter  $k$  for constant  $d$ , which can be seen by reduction to  $(d + 2)$ -HITTING SET. For  $d = 1$  there is a  $15k$ -vertex kernel and a  $O(2^k \cdot k^2 \cdot kn)$  algorithm. An  $O(k^{1+\epsilon})$ -vertex kernel was shown for  $d \leq 2$ . Furthermore, there is a fixed-parameter algorithm with running time  $O((d + 2)^k + n(k + d))$ . For unbounded  $d$ , BOUNDED DEGREE DELETION is W[2]-complete for the parameter  $k$ . In the following we start with investigating the parameterized complexity with respect to the parameters that measure the “degree of acyclicity” beginning with the “size of a feedback edge set”. The parameterized complexity for “treewidth of the input graph” or “size of a feedback vertex set” remains open in this work.

### 6.2 Size of a feedback edge set as parameter

In this section, we show fixed-parameter tractability for BOUNDED DEGREE DELETION with respect to the parameter  $s_e :=$  “size of a feedback edge set”. A first step will be to show that an annotated version is polynomial-time solvable on acyclic graphs.

**An annotated version on acyclic graphs.** In the following, we suppose that the input graph is acyclic. We describe an algorithm that computes an *optimal solution set*, that is, a vertex subset of minimum size whose removal from the graph is a graph in which each vertex has degree at most  $d$ . To finally solve BOUNDED DEGREE DELETION on graphs with bounded feedback edge set size, we introduce a slightly modified version of BOUNDED DEGREE DELETION and show that it is solvable in polynomial time on acyclic graphs. The modified problem is defined as follows:

# ANNOTATED BOUNDED DEGREE DELETION

*Given:* An undirected graph  $G = (V, E)$ , a vertex subset  $U$ , and integers  $d \geq 0$  and  $k \geq 0$ .

*Question:* Does there exist a subset  $V' \subseteq (V \setminus U)$  of size at most  $k$  whose removal from  $G$  yields a graph in which each vertex has degree at most  $d$ ?

The vertex subset  $U$  is called the set of *unremovable* vertices. The algorithm uses a specialized bottom-up tree-traversal and handles each vertex exactly once: Either the vertex will be marked to be “not contained in the solution set” or the vertex will be removed. This process is called *decision step* of the algorithm. The order of processing the vertices corresponds to their depth<sup>1</sup> (from higher to lower). Vertices that have the same depth are handled in order of their degree (from higher to lower). This ensures three invariants in the decision step for vertex  $x$ :

1. Each child of  $x$  was either removed or was marked.
2. Each child of every sibling of  $x$  was either removed or was marked.
3. There is no sibling of  $x$  with higher degree which was not already removed or marked.

The decision step of the algorithm is given in Figure 6.1. It is easy to see that after each decision step for a vertex  $x$  either:

- $x$  was removed, or
- $x$  was marked and has degree at most  $d$ , or
- the algorithm canceled due to the detection of a no-instance.

Let  $M$  be the set of marked vertices and  $S := V \setminus M$ .

**Lemma 12.** *The algorithm computes an optimal solution set in  $O(n^2)$  time.*

*Proof.* It remains to show that if the decision step was correct and optimal for each child of a vertex  $x$ , then the decision step is also correct and optimal for  $x$ . Due to the processing order, the correctness (and optimality) of the whole algorithm follows. This proof is more or less a complete induction: The base clause is “the decision step is correct and optimal for each child of a leaf” which is clearly given, because a leaf has no child. In the following we show correctness and optimality for

---

<sup>1</sup>The depth of a vertex  $x$  in a tree is the length of the path between  $x$  and the root.



**Decision step**

**Input:** An undirected acyclic graph  $G$  and a vertex  $x$  from  $G$ .

**Require:** Each child of  $x$  was either removed or was marked. Each child of every sibling of  $x$  was either removed or was marked. There is no sibling of  $x$  with higher degree which was not already removed or marked.

Let  $p$  denote the parent vertex of  $x$  and let  $p_p$  denote the parent of  $p$ .

**Case A**  $x$  is unremovable

**Case A.1** If  $\deg(x) = d + 1$  and  $p$  is removable, then remove  $p$  and mark  $x$  to be “not contained in the solution set”.

**Case A.2** If  $\deg(x) = d + 1$  and  $p$  is unremovable, then cancel and return “no”.

**Case A.3** If  $\deg(x) > d + 1$ , then cancel and return “no”.

**Case A.4** If  $\deg(x) < d + 1$ , then mark  $x$  to be “not contained in the solution set”.

**Case B**  $x$  is removable**Case B.1**  $p$  is removable

**Case B.1.a** If  $\deg(x) < d + 1$ , then mark  $x$  to be “not contained in the solution set”.

**Case B.1.b** If  $\deg(x) > d + 1$ , then remove  $x$ .

**Case B.1.c** If  $\deg(x) = d + 1$ , then remove the parent vertex of  $x$ . If there is no parent vertex ( $x$  is the root), then just remove  $x$ .

**Case B.2**  $p$  is unremovable

**Case B.2.a** If  $\deg(x) \geq d + 1$ , then remove  $x$ .

**Case B.2.b** If  $\deg(x) < d + 1$  and  $\deg(p) < d + 1$ , then mark  $x$  to be “not contained in the solution set”.

**Case B.2.c** It holds that  $\deg(x) < d + 1$  and  $\deg(p) \geq d + 1$ . If  $p_p$  exists and  $p_p$  is removable, then remove  $p_p$ . If  $\deg(p) \geq d + 1$  (possibly after removing  $p_p$ ), then remove  $x$ .

**Ensure:** Either  $x$  is removed, or  $x$  is marked and has degree at most  $d$ , or the algorithm cancels due to the detection of a no-instance. Furthermore, the decision is optimal with respect to the total number of removed vertices.

Figure 6.1: Decision step of the algorithm. The parent vertex of  $x$  is denoted as  $p$ .

each case:

- Case A.1 Since the decision was correct for each child,  $p$  is the only neighbor of  $x$  that can be removed. Removing  $x$  is not allowed due to the problem definition. To decrease the degree of  $x$  to  $d$  one must remove  $p$ .
- Case A.2 No neighbor of  $x$  can be removed. Hence, it is not possible to decrease the degree of  $x$ .
- Case A.3 Analogously to case A.1,  $p$  is the only neighbor of  $x$  that can be removed. Removing  $x$  is not allowed due to the problem definition. Thus, it is not possible to decrease the degree of  $x$  by more than one anyway.
- Case A.4 This case is correct by the problem definition.
- Case B.1.a Removing  $x$  does not lead to an optimal solution. The degree of  $x$  is already small enough and  $p$  is removable. Hence, it is at least as good to remove  $p$  instead of  $x$ .
- Case B.1.b Clearly, one must remove either  $x$  or at least 2 neighbors of  $x$ . Since the decision was correct for each child,  $p$  is the only neighbor of  $x$  that can be removed. Hence, one must remove  $x$  anyway.
- Case B.1.c Clearly, one must remove either  $x$  or  $p$ . It is not necessary to remove  $x$ , because removing  $p$  instead is always better. Note that the degree of each child of  $x$  is already at most  $d$ .
- Case B.2.a No neighbor of  $x$  can be removed. One must decrease the degree of  $x$  at least by one. Clearly, the only way to do this is removing  $x$ .
- Case B.2.b Removing  $x$  does not lead to an optimal solution. The degree of  $x$  and the degree of the only neighbor  $p$  is already small enough so that we do not need to remove  $x$ . Hence, not removing  $x$  cannot be wrong.
- Case B.2.c One must remove enough neighbors of  $p$  such that it has final degree at most  $d$ . Due to the invariants, no remaining sibling has another neighbor than  $p$  and no sibling has degree more than  $d$ . More precisely,  $x$  has degree at most  $d$  and siblings with higher degree were processed before. It does not matter which child of  $p$  will be removed, but removing the parent vertex can possibly decrease the degree of another vertex.

Hence, the vertex subset  $S$  is an optimal solution set. It remains to prove the running time. Collecting the degree information for each vertex and ordering the vertices according to their depth and degree takes  $O(n^2)$  time. Each decision step is computable in  $O(n)$  time: Marking a vertex or checking whether a vertex is marked takes constant time. Removing a vertex while updating the degree information and updating the ordering of the vertices takes  $O(n)$  time. Since each vertex is only processed once, the algorithm needs  $n$  decision steps. Thus, the overall running time is in  $O(n^2)$ .  $\square$

**Extension to the case of bounded feedback edge set size.** Let  $(G, d, k)$  be an instance of BOUNDED DEGREE DELETION. In the following, we assume that there is a feedback edge set  $E_f$  of size  $s_e$ . The first step is a search tree that transforms

the instance into acyclic instances. We branch on the feedback edge set elements into three cases. Let  $\{x, y\}$  be an edge in  $E_f$ :

**Branching case 1** Remove  $x$ .

**Branching case 2** Remove  $y$ .

**Branching case 3** Do not remove  $x$  and  $y$ . Instead, remove  $\{x, y\}$  and add one additional leaf  $a_x$  to  $x$  and one additional leaf  $a_y$  to  $y$ . (This is necessary to preserve the degrees.) Mark  $x$ ,  $a_x$ ,  $y$ , and  $a_y$  as “unremovable”.

Let  $G'$  be the resulting graph,  $U$  be the set of vertices that are marked as “unremovable”, and  $r$  the number of removed vertices. In each search tree leaf  $G'$  is acyclic. We have to guarantee to find an optimal solution set in at least one branch. Thus, a second step is to solve the ANNOTATED BOUNDED DEGREE DELETION instance  $(G', U, d, k - r)$  in each search tree leaf. If the ANNOTATED BOUNDED DEGREE DELETION instance in any search tree leaf is a yes-instance, then return “yes”. Otherwise, return “no”. Putting all together, we arrive at the following:

**Theorem 13.** BOUNDED DEGREE DELETION can be solved in  $O(3^{s_e} \cdot n^2)$  time with  $s_e$  being the size of a feedback edge set.

*Proof.* The running time of the algorithm is clearly in  $O(3^{s_e} \cdot n^2)$ . Furthermore, it is easy to see that if there is a yes-instance of ANNOTATED BOUNDED DEGREE DELETION in a search tree leaf, then the original BOUNDED DEGREE DELETION instance is a yes-instance, too. It remains to show that if the original instance is a yes-instance of BOUNDED DEGREE DELETION, then there is a yes-instance of ANNOTATED BOUNDED DEGREE DELETION in at least one search tree leaf. Assume, that there is a solution set  $M_d$  for the original instance  $(G, d, k)$ . Let  $V_I$  denote the vertices that are incident to an edge from  $E_f$ . Let  $M_I := V_I \cap M_d$  denote the solution set vertices that are incident to an edge from  $E_f$ . Let  $M_I^2 := \{x \mid x \in M_I \wedge \exists y : y \in M_I \wedge \{x, y\} \in E_f\}$  denote the solution set vertices that are connected to another solution set vertex by an edge from  $E_f$ . It is easy to verify that there is a search tree leaf  $s$  with  $U_s$  being the set of as “unremovable” marked vertices and  $R_s$  is the set of removed vertices such that:

- For each vertex  $x \in M_I^2$  either  $x \in R_s$  and  $y \notin U_s$  or  $y \in R_s$  and  $x \notin U_s$ .
- For each vertex  $x \in (M_I \setminus M_I^2)$  it holds that  $x \in R_s$ .
- Each vertex  $u \in U_s$  is not part of the solution set  $M_d$ .

Clearly, the ANNOTATED BOUNDED DEGREE DELETION instance in the search tree leaf  $s$  is a yes-instance.  $\square$



## 7 Conclusion and Outlook

We investigated the parameterized complexity of three similar vertex deletion problems. Since each problem is solvable in polynomial time when restricted to acyclic graphs, but NP-hard in general, it was natural to study fixed-parameter tractability with respect to parameters that measure the “degree of acyclicity”. More precisely, we considered  $s_e$  := “size of a feedback edge set” respectively  $s_a$  := “size of a feedback arc set”,  $s_v$  := “size of a feedback vertex set”, and  $t_w$  := “treewidth of the input graph”.

For MINIMUM INDEGREE DELETION, which is equivalent to constructive control by deleting candidates in Llull elections, we showed that it is W[2]-hard with respect to the parameters  $s_a$  and  $s_v$ . In addition, it is at least in XP with respect to  $s_v$  (and  $s_a$ ). Although it is W[2]-complete with respect to the parameter  $k$  := “solution size”, we could show fixed-parameter tractability with respect to the combined parameters  $(s_a, k)$  and  $(s_v, k)$ . The fixed-parameter algorithm in Section 4.3 solves MINIMUM INDEGREE DELETION in  $O(s_v \cdot (k+1)^{s_v} \cdot n^2)$  time. There should be room for improvements of this algorithm.

In contrast to MINIMUM INDEGREE DELETION, MINIMUM DEGREE DELETION is even fixed-parameter tractable with respect to each of the parameters that measure the “degree of acyclicity”. We showed that MINIMUM DEGREE DELETION has no polynomial kernel with respect to  $s_v$  or  $t_w$ , unless the polynomial-time hierarchy collapses. Moreover, there is also no polynomial kernel with respect to the combined parameter  $(s_c^*, k)$  with  $s_c^*$  := “size of a vertex cover that does not contain  $w_c$ ”. However, a vertex-linear kernel was found with respect to  $s_e$ . In future research it would be desirable to develop a practicable algorithm with respect to  $t_w$ . A first step towards this was done in Section 5.4, where a fixed-parameter algorithm with respect to  $s_v^*$  := “size of a feedback vertex set that does not contain  $w_c$ ” using dynamic programming was developed. Possibly, there are ways to adapt the ideas to the parameter  $s_v$ .

For BOUNDED DEGREE DELETION we showed a fixed-parameter algorithm using a depth-bounded search tree with running time  $O(3^{s_e} \cdot n^2)$ . Improving this algorithm (for example by data reduction rules) or developing a fixed-parameter algorithm for the combined parameter  $(s_v, k)$  are promising future tasks. In addition, the parameterized complexity of BOUNDED DEGREE DELETION with respect to the parameters  $s_v$  and  $t_w$  still remains open.



# Bibliography

- [ALS91] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.
- [BCK<sup>+</sup>05] Nicole E. Baldwin, Elissa J. Chesler, Stefan Kirov, Michael A. Langston, Jay R. Snoddy, Robert W. Williams, and Bing Zhang. Computational, integrative, and comparative methods for the elucidation of genetic co-expression networks. *Journal of Biomedicine and Biotechnology*, 2:172–180, 2005.
- [BDFH09] Hans Bodlaender, Rodney Downey, Michael Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.
- [BDHK06] Dietmar Berwanger, Anuj Dawar, Paul Hunter, and Stephan Kreutzer. Dag-width and parity games. In *Proceedings of 23rd Annual Symposium on Theoretical Aspects of Computer Science*, volume 3884 of *Lecture Notes in Computer Science*, pages 524–536. Springer, 2006.
- [Bel03] Richard Ernest Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 2003.
- [BFH<sup>+</sup>08] Eric Brelsford, Piotr Faliszewski, Edith Hemaspaandra, Henning Schnoor, and Ilka Schnoor. Approximability of manipulating elections. In *AAAI’08: Proceedings of the 23rd National Conference on Artificial Intelligence*, pages 44–49. AAAI Press, 2008.
- [BGK08] Hans L. Bodlaender, Alexander Grigoriev, and Arie M. C. A. Koster. Treewidth lower bounds with brambles. *Algorithmica*, 51(1):81–98, 2008.
- [Bod06] Hans L. Bodlaender. Treewidth: Characterizations, applications, and computations. In *Proceedings of Graph-Theoretic Concepts in Computer Science, 32nd International Workshop*, pages 1–14, 2006.
- [Bod08] Hans L. Bodlaender. Treewidth of graphs. In *Encyclopedia of Algorithms*, pages 968–982. Springer, 2008.
- [BPT92] Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(5&6):555–581, 1992.

- [BTT89] John J. Bartholdi, Craig A. Tovey, and Michael A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6:227–241, 1989.
- [BTT92] John J. Bartholdi, Craig A. Tovey, and Michael A. Trick. How hard is it to control an election? *Mathematical and Computer Modelling*, 16(8-9):27–40, 1992.
- [BTY08] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Analysis of data reduction: Transformations give evidence for non-existence of polynomial kernels. Technical Report UU-CS-2008-030, Department of Information and Computing Sciences, Utrecht University, 2008.
- [BU09] Nadja Betzler and Johannes Uhlmann. Parameterized complexity of candidate control in elections and related digraph problems. *Theoretical Computer Science*, 410(52):5425–5442, 2009.
- [CCHO05] Jin-Yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogiwara. Competing provers yield improved Karp-Lipton collapse results. *Information and Computation*, 198(1):1–23, 2005.
- [CELM07] Yann Chevaleyre, Ulle Endriss, Jérôme Lang, and Nicolas Maudet. A short introduction to computational social choice. In *Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science*, volume 4362 of *Lecture Notes In Computer Science*, pages 51–69. Springer, 2007.
- [CFL<sup>+</sup>08] Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences*, 74(7):1188–1198, 2008.
- [CFRS07] Robin Christian, Mike Fellows, Frances Rosamond, and Arkadii Slinko. On complexity of lobbying in multiple referenda. *Review of Economic Design*, 11(3):217–224, November 2007.
- [CLL<sup>+</sup>08] Jianer Chen, Yang Liu, Songjian Lu, Barry O’sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM*, 55(5):1–19, 2008.
- [CLS<sup>+</sup>05] Elissa J. Chesler, Lu Lu, Siming Shou, Yanhua Qu, Jing Gu, Jintao Wang, Hui Chen Hsu, John D. Mountz, Nicole E. Baldwin, Michael A. Langston, David W. Threadgill, Kenneth F. Manly, and Robert W. Williams. Complex trait analysis of gene expression uncovers polygenic and pleiotropic networks that modulate nervous system function. *Nature Genetics*, 37(3):233–42, 2005.
- [Cou90] Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B)*, pages 193–242. 1990.



- 
- [Cou09] Bruno Courcelle. *Graph Structure and Monadic Second Order Logic*. Cambridge University Press, In preparation, 2009.
- [CSL07] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. When are elections with few candidates hard to manipulate. *Journal of the ACM* 54, 54(3):1–33, 2007.
- [DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.
- [DLS09] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through colors and ids. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, volume 5555 of *Lecture Notes In Computer Science*, pages 378–389. Springer, 2009.
- [DLSV08] Michael Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Capacitated domination and covering: A parameterized perspective. In *Proceedings of Parameterized and Exact Computation, Third International Workshop*, volume 5018 of *Lecture Notes in Computer Science*, pages 78–90. Springer, 2008.
- [DS05] Irit Dinur and Shmuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.
- [ENSS98] Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.
- [FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer, 2006.
- [FHHR09a] Piotr Faliszewski, Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Llull and copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence Research*, 35(1):275–341, 2009.
- [FHHR09b] Piotr Faliszewski, Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. A richer understanding of the complexity of election systems. In *Fundamental Problems in Computing: Essays in Honor of Professor Daniel J. Rosenkrantz*, pages 375–406. Springer, 2009.
- [FS08] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 133–142. ACM, 2008.
- [Fuj98] Toshihiro Fujito. A unified approximation algorithm for node-deletion problems. *Discrete Applied Mathematics*, 86(2-3):213–231, 1998.

- [HHR05] Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Anyone but him: the complexity of precluding an alternative. In *Proceedings of the 20th National Conference on Artificial Intelligence*, volume 1 of *Aaai Conference On Artificial Intelligence*, pages 95–101. AAAI Press, 2005.
- [HK07] Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 637–644. ACM, 2007.
- [JRST01] Thor Johnson, Neil Robertson, Paul D. Seymour, and Robin Thomas. Directed tree-width. *Journal of Combinatorial Theory Series B*, 82(1):138–154, 2001.
- [Kan60] Leonid V. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, 6(4):366–422, 1960.
- [Kan87] Ravi Kannan. Minkowski’s convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.
- [Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
- [Len83] Hendrik Willem Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.
- [LY80] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.
- [MATV10] Daniel Meister, Jan Arne Telle, and Martin Vatshelle. Recognizing digraphs of Kelly-width 2. *Discrete Applied Mathematics*, 158(7):741–746, 2010.
- [Mos10] Hannes Moser. *Finding Optimal Solutions for Covering and Matching Problems*. Cuvillier, E, 1., Aufl. edition, 1 2010.
- [MPRZ08] Reshef Meir, Ariel D. Procaccia, Jeffrey S. Rosenschein, and Aviv Zohar. Complexity of strategic behavior in multi-winner elections. *Journal of Artificial Intelligence Research*, 33:149–178, 2008.
- [Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
- [Nie10] Rolf Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proceedings of 27th International Symposium on Theoretical Aspects of Computer Science*, volume 5 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17–32. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

- 
- [OB03] Michael Okun and Amnon Barak. A new approach for approximating node deletion problems. *Information Processing Letters*, 88(5):231–236, 2003.
- [Obd06] Jan Obdržálek. Dag-width: connectivity measure for directed graphs. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 814–821. ACM, 2006.
- [PY91] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.
- [RS86] Neil Robertson and Paul D. Seymour. Graph minors II: algorithmic aspects of tree-width. *Journal Algorithms*, 7:309–322, 1986.
- [Saf05] Mohammad Ali Safari. D-width: A more natural measure for directed tree width. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science*, volume 3618 of *Lecture Notes in Computer Science*, pages 745–756. Springer, 2005.
- [Sto76] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.
- [Yap83] Chee K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26(3):287–300, 1983.



# Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Jena, den 25. Mai 2010

Robert Bredereck